






Public repository

Branches Tags

☰

README.md

✎

react-native-tableview-simple

Donate

Patreon

build

unknown

npm

v4.4.0

downloads

3.9k/month

quality

★★★★★

Snyk security

monitored

license

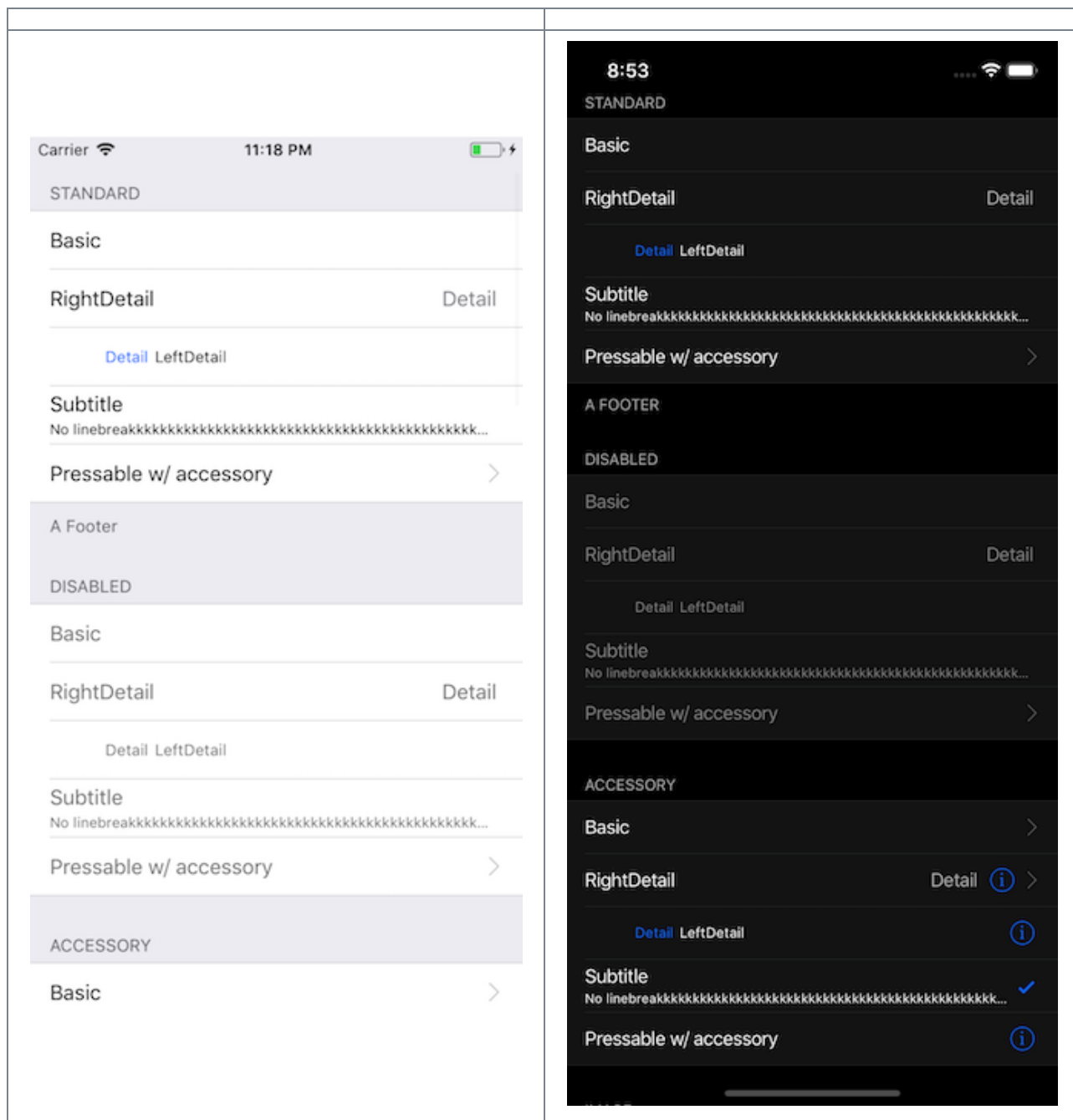
MIT

👉 *This component is used in my production app **Game ideas**. Make sure to check it out!* 👉

This cross-platform component is inspired by the iOS-TableView. Made with pure CSS, the intention is to provide a **flexible and lightweight alternative to a bridged component**. Don't be scared of React-Native upgrades anymore!

A possible use case might be an about- or a settings-screen with a few rows. For displaying long datalists it is recommended to use the `FlatList` Component together with `cell` and `Separator` Components. (see [example](#) or [live demo with expo](#))

🚀 If you like my component and want to buy me a coffee press the [sponsor](#) Button and find out about GitHub Sponsors – Thanks! 👉



Have a look at the [examples below!](#) :-)

- [Installation](#)
- [Extensible](#)
- [Props: TableView Cell Section Separator](#)
- [Examples](#)
- [Try it out](#)

Installation

1. Install as dependency:

```
// yarn
yarn add react-native-tableview-simple
// or npm
npm i react-native-tableview-simple --S
```



2. Add needed components:

```
import { Cell, Section, TableView } from 'react-native-tableview-simple';
```



Extensible

`react-native-tableview-simple` provides you with some predefined CSS-styles, inspired by the native `UITableView`. You can always mix the `Cell` -instances inside a `Section`, with other (React-Native)-Views.

Override defaults of `Cell` -Component

Don't repeat yourself. If you override the default props over and over again, just pass them as an object.

```
const cellPropsCustom = {
  cellStyle: 'Basic',
  title: 'Basic Custom',
  backgroundColor: 'black',
};
```



```
<Cell onPress={console.log} {...cellPropsCustom} />
<Cell onPress={console.log} {...cellPropsCustom} />
```

Separator BackgroundColor is derived from Cell BackgroundColor

The `Separator` -Component is a line from the left end to the right end. According to the original iOS `UITableView` there should be an inset on the left end. This is done by separating the `Separator` -Component in two parts: `SeparatorContainer` (full width) and `SeparatorInner` (width - inset). (See: [Separator.tsx](#)) The `SeparatorContainer` has the same color that the `cell` -Component above. The `SeparatorInner` has the default Separator Color. Pressing a Cell Component will change the color of `SeparatorInner` to `transparent` .

Why is that so complicated?

Because just hiding the separator would make the height of the component jump.

Props

- [TableView](#)
- [Cell](#)
- [Section](#)
- [Separator](#)

TableView

The `TableView` component controls the theme.

| Prop | Default | Type | Description |
|-------------------|---------|------------------|---|
| children | - | React.ReactNode | Children. Should be of type <code>Section</code> |
| appearance | auto | string | <code>auto</code> : System Appearance; <code>light</code> : Light Appearance; <code>dark</code> : Dark Appearance; <code>[string]</code> : Appearance defined through <code>customAppearances</code> |
| customAppearances | - | THEME_APPEARANCE | |
| style | - | ViewStyle | Applied to the table wrapper |

Section

The `Section` component is needed to render the `cells` together with `Separators`. It's possible to use the `FlatList` component instead ([Example](#)).

| Prop | Default | Type | Description |
|--|----------------------|-----------------------------------|---|
| <code>allowFontScaling</code> | <code>true</code> | <code>bool</code> | Respect Text Size accessibility setting iOS |
| <code>footerComponent</code> | <code>-</code> | <code>React.Component</code> | Inject any compon replace original fo (optional) |
| <code>headerComponent</code> | <code>-</code> | <code>React.Component</code> | Inject any compon replace original he (optional) |
| <code>footer</code> | <code>-</code> | <code>string</code> | Footer value |
| <code>footerTextColor</code> | <code>#6d6d72</code> | <code>string</code> | Text color of foote |
| <code>footerTextStyle</code> | <code>{}</code> | <code>Text.propTypes.style</code> | These styles will be applied to the foot <code>Text</code> -Component |
| <code>header</code> | <code>-</code> | <code>string</code> | Header value |
| <code>headerTextColor</code> | <code>#6d6d72</code> | <code>string</code> | Text color of head |
| <code>headerTextStyle</code> | <code>{}</code> | <code>Text.propTypes.style</code> | These styles will be applied to the hea <code>Text</code> -Component |
| <code>hideSeparator</code> | <code>false</code> | <code>bool</code> | Hide separators |
| <code>hideSurroundingSeparators</code> | <code>false</code> | <code>bool</code> | Hide surrounding separators, best cc with <code>roundedCorners</code> |
| <code>roundedCorners</code> | <code>false</code> | <code>bool</code> | Enable rounded cc best combined wit <code>hideSurroundingSe</code> |
| <code>sectionPaddingBottom</code> | <code>15</code> | <code>number</code> | Padding bottom o section |
| <code>sectionPaddingTop</code> | <code>15</code> | <code>number</code> | Padding top of sec |

| Prop | Default | Type | Description |
|---------------------|--------------------------------|--------|--|
| sectionTintColor | #EFEFF4 | string | Background color section |
| separatorInsetLeft | 15 | number | Left inset of separator |
| separatorInsetRight | 0 | number | Right inset of separator |
| separatorTintColor | #C8C7CC | string | Color of separator |
| withSafeAreaView | true / false (on iOS <= 10) | bool | Render section header and footer with SafeAreaView |

Cell

The style of the `cell` component is inspired by the native `UITableViewCell`. Because the `cell` component is created with CSS only, its highly flexible. The content of the cell is separated in three views, which can all be modified via `props`: `cellImageView` | `cellContentView` | `cellAccessoryView`.

To get an idea what you can modify via `props`, have a look at the [examples below](#).

| Prop | Default | Type | Description |
|-----------------------------------|---------|--------|--|
| accessory | - | string | Predefined Disclosure Indicator, DisclosureDetail, DisclosureDetailDisclosure, DisclosureCheckmark |
| accessoryColor | #007AFF | string | Color of a |
| accessoryColorDisclosureIndicator | #C7C7CC | string | Color of a Disclosure |
| allowFontScaling | true | bool | Respect Text accessibility on iOS |
| backgroundColor | #FFF | string | Background color of cell |

| Prop | Default | Type | Desc |
|-----------------------|---------|----------------------|---|
| cellStyle | Basic | string | Predefined styles: Basic , RightDetail, LeftDetail, Subtitle |
| cellAccessoryView | - | React.Component | Replace a cell accessory view component <i>add Switch, ActivityIndicator</i> |
| cellContentView | - | React.Component | Replace cell content component |
| cellImageView | - | React.Component | Replace cell image component |
| children | - | React.Component | Additional components to be displayed in the cell. (e.g. <i>Picker</i> or <i>DateTimePicker</i>) |
| contentContainerStyle | {} | View.propTypes.style | These styles are applied to content container which wraps each child view <i>cellStyle</i> <i>Override padding and padding set fixed height</i> |
| detail | - | string OR number | Detail value |
| detailTextProps | {} | Text.propTypes | These props are applied to (right-) detail component |
| detailTextStyle | {} | Text.propTypes.style | These styles are applied to |

| Prop | Default | Type | Desc |
|------------------------|-----------|--|--|
| | | | right-) detail Component |
| disableImageResize | false | bool | Disable re image |
| hideSeparator | false | bool | Hide the f Separator Component |
| highlightActiveOpacity | 0.8 | number | Opacity o touch is a |
| highlightUnderlayColor | black | string | Color of u will show when touc |
| isDisabled | false | bool | Cell is disa onPress triggered |
| image | - | React.Component (Image) | Image cor displayed Will resize automatic |
| leftDetailColor | #007AFF | string | Text color detail |
| rightDetailColor | #8E8E93 | string | Text color detail |
| subtitleColor | #000 | string | Text color |
| subtitleTextStyle | {} | Text.propTypes.style | These styl applied to Text -Col |
| testID | undefined | string | Add testID compone |
| title | - | string OR number OR React.Component | Title value |

| Prop | Default | Type | Desc |
|------------------------|-----------------------------|----------------------|---|
| titleTextColor | #000 | string | Text color |
| titleTextProps | {} | Text.propTypes | These props applied to Text -Component |
| titleTextStyle | {} | Text.propTypes.style | These styles applied to Text -Component (e.g.: update fontSize, fontFamily) |
| titleTextStyleDisabled | {} | Text.propTypes.style | These styles applied to Text -Component when the cell is disabled |
| onPress | - | func or false | If set, cell will be automatically initialized with TouchableOpacity |
| onPressDetailAccessory | - | func or false | Listen to onPress event of cell detail accessory |
| withSafeAreaView | true / false (on iOS <= 10) | bool | Render cell with SafeAreaView |

Wrap Cell

Sometimes custom cell components are needed. By creating new component, which is based on cell , its only necessary to set the props once.

...

import {



```
Cell,
Section,
TableView,
} from 'react-native-tableview-simple';

const CellVariant = (props) => (
  <Cell
    {...props}
    cellContentView={
      <View
        style={{ alignItems: 'center', flexDirection: 'row', flex: 1, paddingVertica
      >
        <Text
          allowFontScaling
          numberOfLines={1}
          style={{ flex: 1, fontSize: 20 }}
        >
          {props.title}
        </Text>
      </View>
    }
  />
);

...

<TableView>
  <Section>
    <CellVariant title="Element 1" />
    <CellVariant title="Element 2" />
    <CellVariant title="Element 3" />
    <CellVariant title="Element 4" />
  </Section>
</TableView>

...
```

Separator

In general the `Separator` component is used internally by the `Section` component. But additionally this component can be used together with `FlatList` . See the [example below](#).

| Prop | Default | Type | Description |
|-----------------|---------|--------|------------------|
| backgroundColor | #EFEFF4 | string | Background color |

| Prop | Default | Type | Description |
|------------------|-----------------------------|--------|---------------------------------------|
| insetLeft | 15 | number | Left inset of separator |
| insetRight | 0 | number | Right inset of separator |
| isHidden | false | bool | Hide separator but keeping its height |
| tintColor | #C8C7CC | string | Color of separator |
| withSafeAreaView | true / false (on iOS <= 10) | bool | Render separator with SafeAreaView |

Examples

The following examples can be found in the folder `example`. To run the example project, follow these steps:

1. `git clone https://github.com/Purii/react-native-tableview-simple`
2. `cd example`
3. `yarn` OR `npm i`
4. run `/example/ios/example.xcodeproj` via Xcode

- [Quick look](#)
- [Use case: About-screen](#)
- [Complete example / vs. native iOS](#)
- [Render with FlatList](#)

Quick look

```
// ActivityIndicator as accessory
<Cell
  title="Switch"
  cellAccessoryView={<Switch />}
  contentContainerStyle={{ paddingVertical: 4 }} // Adjust height
/>

// Switch as accessory
<Cell
  title="ActivityIndicator"
  cellAccessoryView={<ActivityIndicator />}
/>
```



```
// TextInput
<Cell
  cellContentView={<TextInput style={{fontSize: 16, flex: 1}} placeholder="TextInp
/>

// Image
<Cell
  title="Image"
  image={
    <Image
      style={{ borderRadius: 5 }}
      source={{
        uri: 'https://facebook.github.io/react/img/logo_og.png',
      }}
    />
  }
/>
```

Use case: About-screen

[Help / FAQ](#)[Contact Us](#)

All rights reserved.

```
/**
 * Sample React Native App
 * https://github.com/facebook/react-native
 * @flow
 */

import React, { Component } from 'react';
import { AppRegistry, ScrollView, StyleSheet, Text, View } from 'react-native';
import { Cell, Section, TableView } from 'react-native-tableview-simple';

export default class App extends Component<{}> {
  render() {
    return (
      <ScrollView contentContainerStyle={styles.stage}>
        <View
          style={{
            backgroundColor: '#37474F',
            height: 500,
```



```

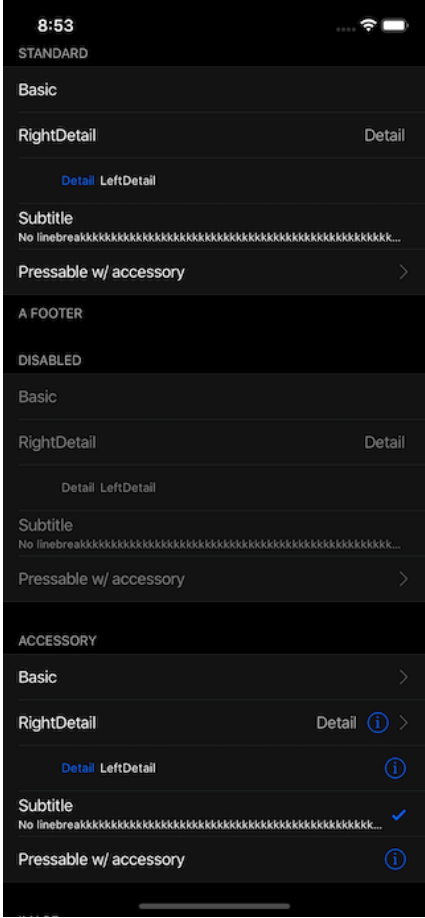
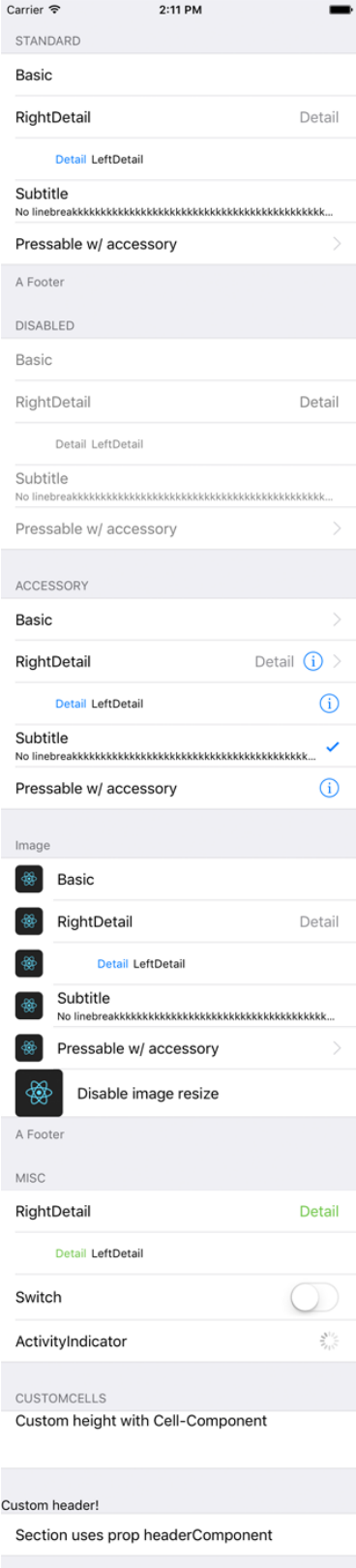
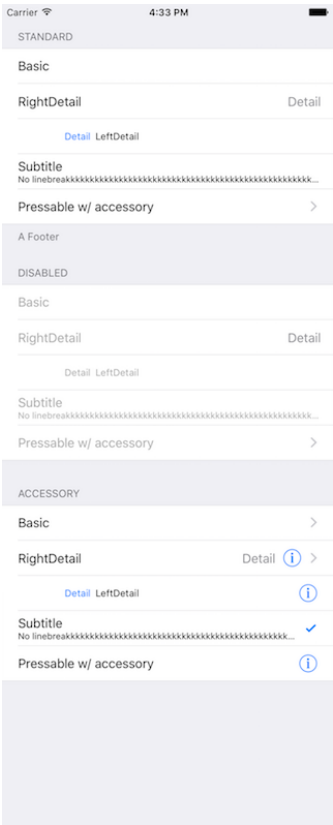
        alignItems: 'center',
        justifyContent: 'center',
      }}>
<View
  style={{
    backgroundColor: '#ffc107',
    width: 80,
    height: 80,
    borderRadius: 10,
  }}
/>
</View>
<TableView>
  <Section footer="All rights reserved.">
    <Cell
      title="Help / FAQ"
      titleTextColor="#007AFF"
      onPress={() => console.log('open Help/FAQ')}
    />
    <Cell
      title="Contact Us"
      titleTextColor="#007AFF"
      onPress={() => console.log('open Contact Us')}
    />
  </Section>
</TableView>
</ScrollView>
);
}
}

const styles = StyleSheet.create({
  stage: {
    backgroundColor: '#EFFFF4',
    paddingBottom: 20,
    flex: 1,
  },
});

```

react-native-tableview-simple vs. Native iOS

The left and middle screens are build using `react-native-tableview-simple` . The right one is native.

| react-native-tableview-simple (Dark Appearance) | react-native-tableview-simple | Native iOS |
|--|---|--|
|  |  |  |

/**
* Sample React Native App



```
* https://github.com/facebook/react-native
* @flow
*/

import React, { Component } from 'react';
import {
  ActivityIndicator,
  AppRegistry,
  Dimensions,
  Image,
  ScrollView,
  StyleSheet,
  Switch,
  Text,
  TextInput,
  View,
} from 'react-native';
import { Cell, Section, TableView } from 'react-native-tableview-simple';

export default class App extends Component<{}> {
  render() {
    return (
      <ScrollView contentContainerStyle={styles.stage}>
        <TableView>
          <Section header="STANDARD" footer="A Footer">
            <Cell cellStyle="Basic" title="Basic" />
            <Cell cellStyle="RightDetail" title="RightDetail" detail="Detail" />
            <Cell cellStyle="LeftDetail" title="LeftDetail" detail="Detail" />
            <Cell
              cellStyle="Subtitle"
              title="Subtitle"
              detail="No linebreakaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
            />
            <Cell
              cellStyle="Basic"
              title="Pressable w/ accessory"
              accessory="DisclosureIndicator"
              onPress={() => console.log('Heyho!')}
            />
          </Section>
          <Section header="DISABLED">
            <Cell cellStyle="Basic" isDisabled title="Basic" />
            <Cell
              cellStyle="RightDetail"
              isDisabled
              title="RightDetail"
              detail="Detail"
            />
            <Cell
```



```
        cellStyle="LeftDetail"  
        isDisabled  
        title="LeftDetail"  
        detail="Detail"  
    />  
  
    <Cell  
      cellStyle="Subtitle"  
      isDisabled  
      title="Subtitle"  
      detail="No linebreakkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk"  
    />  
  
    <Cell  
      cellStyle="Basic"  
      isDisabled  
      title="Pressable w/ accessory"  
      accessory="DisclosureIndicator"  
      onPress={() => console.log('Heyho!')}  
    />  
  
  </Section>  
  <Section header="ACCESSORY">  
    <Cell  
      cellStyle="Basic"  
      accessory="DisclosureIndicator"  
      title="Basic"  
    />  
  
    <Cell  
      cellStyle="RightDetail"  
      accessory="DetailDisclosure"  
      title="RightDetail"  
      detail="Detail"  
    />  
  
    <Cell  
      cellStyle="LeftDetail"  
      accessory="Detail"  
      title="LeftDetail"  
      detail="Detail"  
    />  
  
    <Cell  
      cellStyle="Subtitle"  
      accessory="Checkmark"  
      title="Subtitle"  
      detail="No linebreakkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk"  
    />  
  
    <Cell  
      cellStyle="Basic"  
      accessory="Detail"  
      title="Pressable w/ accessory"  
      onPress={() => console.log('Heyho!')}  
    />
```

[illegible]

```

    }}
  />
}
/>
<Cell
  cellStyle="Basic"
  title="Pressable w/ accessory"
  accessory="DisclosureIndicator"
  onPress={() => console.log('Heyho!')}
  image={
    <Image
      style={{ borderRadius: 5 }}
      source={{
        uri: 'https://facebook.github.io/react/img/logo_og.png',
      }}
    />
  }
/>
<Cell
  cellStyle="Basic"
  title="Disable image resize"
  disableImageResize
  image={
    <Image
      style={{ height: 50, width: 50, borderRadius: 5 }}
      source={{
        uri: 'https://facebook.github.io/react/img/logo_og.png',
      }}
    />
  }
/>
</Section>
<Section header="MISC">
  <Cell
    cellStyle="RightDetail"
    title="RightDetail"
    detail="Detail"
    rightDetailColor="#6cc644"
  />
  <Cell
    cellStyle="LeftDetail"
    title="LeftDetail"
    detail="Detail"
    leftDetailColor="#6cc644"
  />
  <Cell
    cellStyle="Basic"
    title="Switch"
    cellAccessoryView={<Switch />}
  />

```

```

        contentContainerStyle={{ paddingVertical: 4 }}
      />
      <Cell
        cellStyle="Basic"
        title="ActivityIndicator"
        cellAccessoryView={<ActivityIndicator />}
      />
      <Cell
        cellContentView={
          <TextInput
            style={{ fontSize: 16, flex: 1 }}
            placeholder="TextInput"
          />
        }
      />
    </Section>
    <Section header="CUSTOMCELLS">
      <Cell
        onPress={() => console.log('Heyho!')}
        contentContainerStyle={{ alignItems: 'flex-start', height: 60 }}
        cellContentView={
          <Text style={{ flex: 1, fontSize: 16 }}>
            Custom height with Cell-Component
          </Text>
        }
      />
    </Section>
    <Section headerComponent={<CustomSectionHeader />}>
      <Cell cellStyle="Basic" title="Section uses prop headerComponent" />
    </Section>
  </TableView>
  <View
    style={{
      minHeight: Dimensions.get('window').height,
    }}>
    <View
      style={{
        backgroundColor: '#37474F',
        height: 500,
        alignItems: 'center',
        justifyContent: 'center',
      }}>
      <View
        style={{
          backgroundColor: '#ffc107',
          width: 80,
          height: 80,
          borderRadius: 10,
        }}

```

```

    />
  </View>
  <TableView>
    <Section footer="All rights reserved.">
      <Cell
        title="Help / FAQ"
        titleTextColor="#007AFF"
        onPress={() => console.log('open Help/FAQ')}
      />
      <Cell
        title="Contact Us"
        titleTextColor="#007AFF"
        onPress={() => console.log('open Contact Us')}
      />
    </Section>
  </TableView>
</View>
</ScrollView>
);
}
}

const styles = StyleSheet.create({
  stage: {
    backgroundColor: '#EFEFF4',
    paddingTop: 20,
    paddingBottom: 20,
  },
});

```

Render with FlatList

Be aware of the prop `keyboardShouldPersistTaps` if using `ScrollView` or similar components. (See #85)

```

import React from 'react';
import { FlatList } from 'react-native';

import { Cell, Separator, TableView } from 'react-native-tableview-simple';

const data = [
  { id: 1, title: '1' },
  { id: 2, title: '2' },
  { id: 3, title: '3' },
  { id: 4, title: '4' },
];

```



```
export default ExampleWithFlatList = () => (  
  <TableView style={{ flex: 1 }}>  
    <FlatList  
      data={data}  
      keyExtractor={({item, index}) => item.id}  
      renderItem={({ item, separators }) => (  
        <Cell  
          title={item.title}  
          onPress={console.log}  
          onHighlightRow={separators.highlight}  
          onUnHighlightRow={separators.unhighlight}  
        />  
      )}  
      ItemSeparatorComponent={({ highlighted }) => (  
        <Separator isHidden={highlighted} />  
      )}  
    />  
  </TableView>  
);
```

Try it out

Try it in Expo: <https://snack.expo.io/@purii/react-native-tableview-simple>

Releases 17



4.3.1

Latest

on Dec 12, 2021

Sponsor this project



Purii Patrick Böder

patreon.com/purii

Contributors 17





Report repository