

React Developer Tools

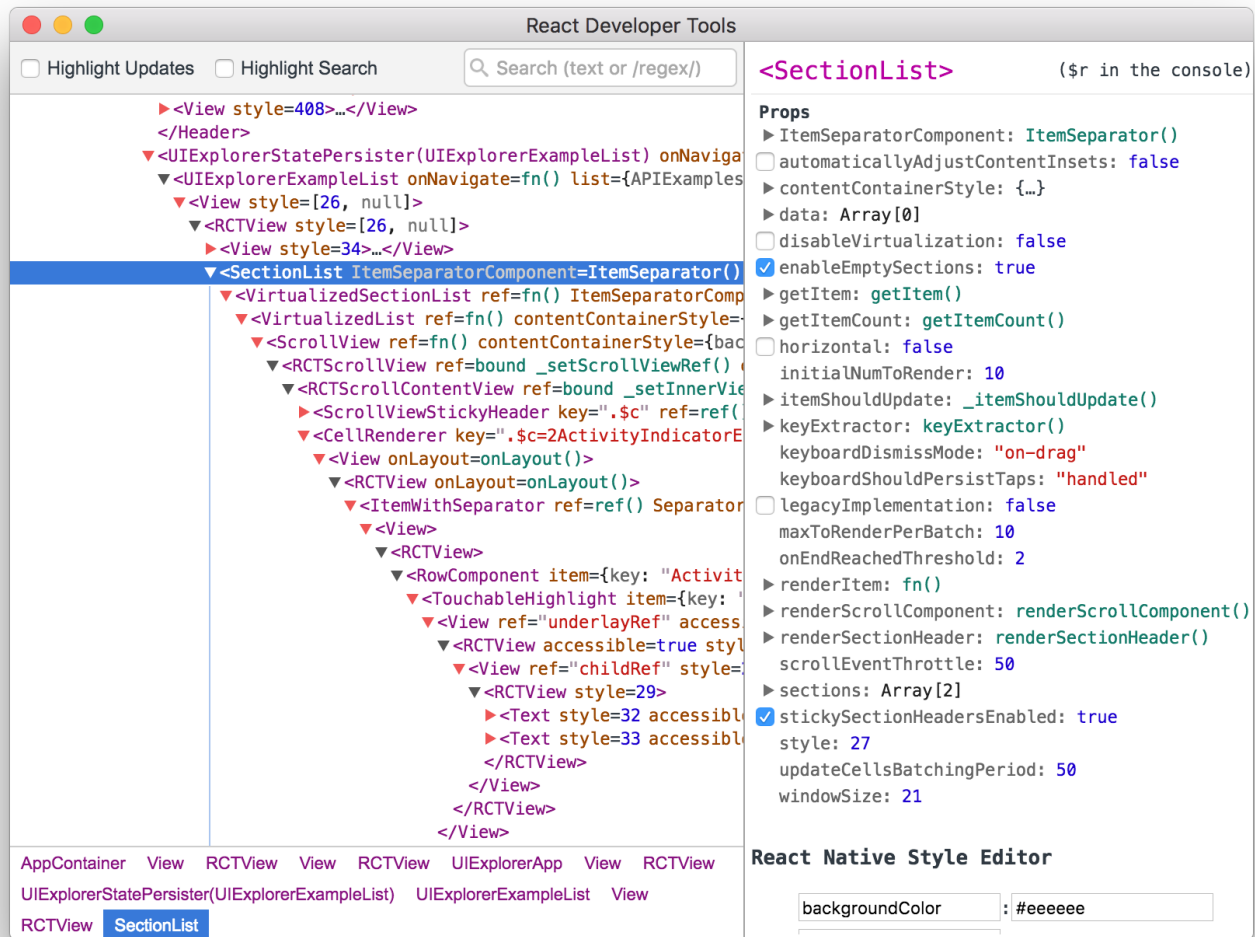
You can use the standalone version of React Developer Tools to debug the React component hierarchy. To use it, install the `react-devtools` package globally:

npm **Yarn**

```
yarn global add react-devtools
```

Now run `react-devtools` from the terminal to launch the standalone DevTools app. It should connect to your simulator within a few seconds.

```
react-devtools
```



! INFO

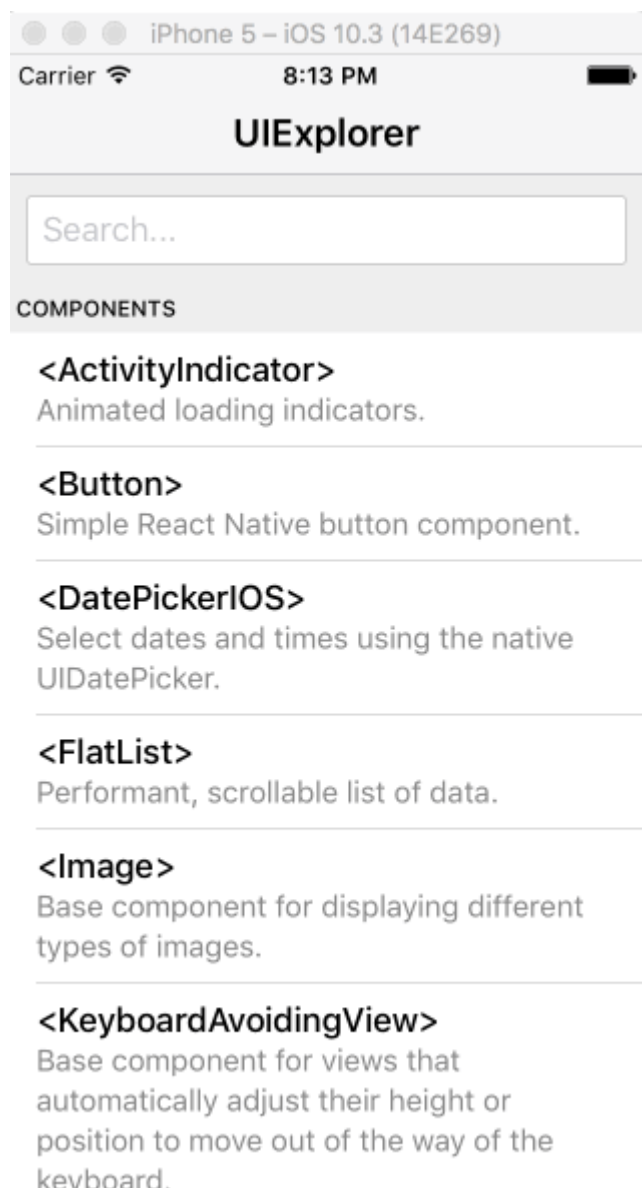
If connecting to the emulator proves troublesome (especially Android 12), try running `adb reverse tcp:8097 tcp:8097` in a new terminal.

! INFO

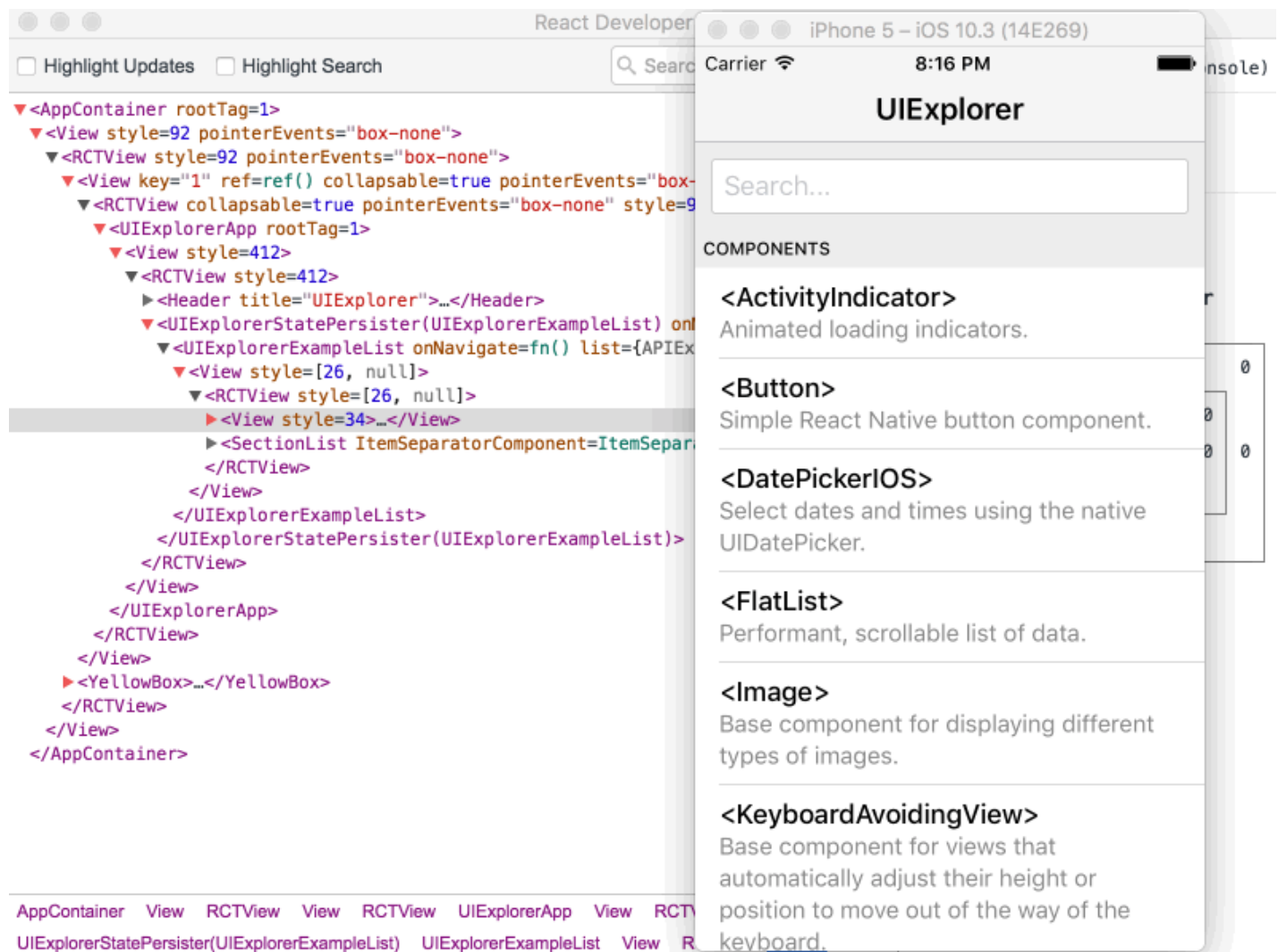
If you prefer to avoid global installations, you can add `react-devtools` as a project dependency. Add the `react-devtools` package to your project using `npm install --save-dev react-devtools`, then add `"react-devtools": "react-devtools"` to the `scripts` section in your `package.json`, and then run `npm run react-devtools` from your project folder to open the DevTools.

Integration with React Native Inspector

Open the Dev Menu and choose "Toggle Inspector". It will bring up an overlay that lets you tap on any UI element and see information about it:



However, when `react-devtools` is running, Inspector will enter a collapsed mode, and instead use the DevTools as primary UI. In this mode, clicking on something in the simulator will bring up the relevant components in the DevTools:



You can choose "Toggle Inspector" in the same menu to exit this mode.

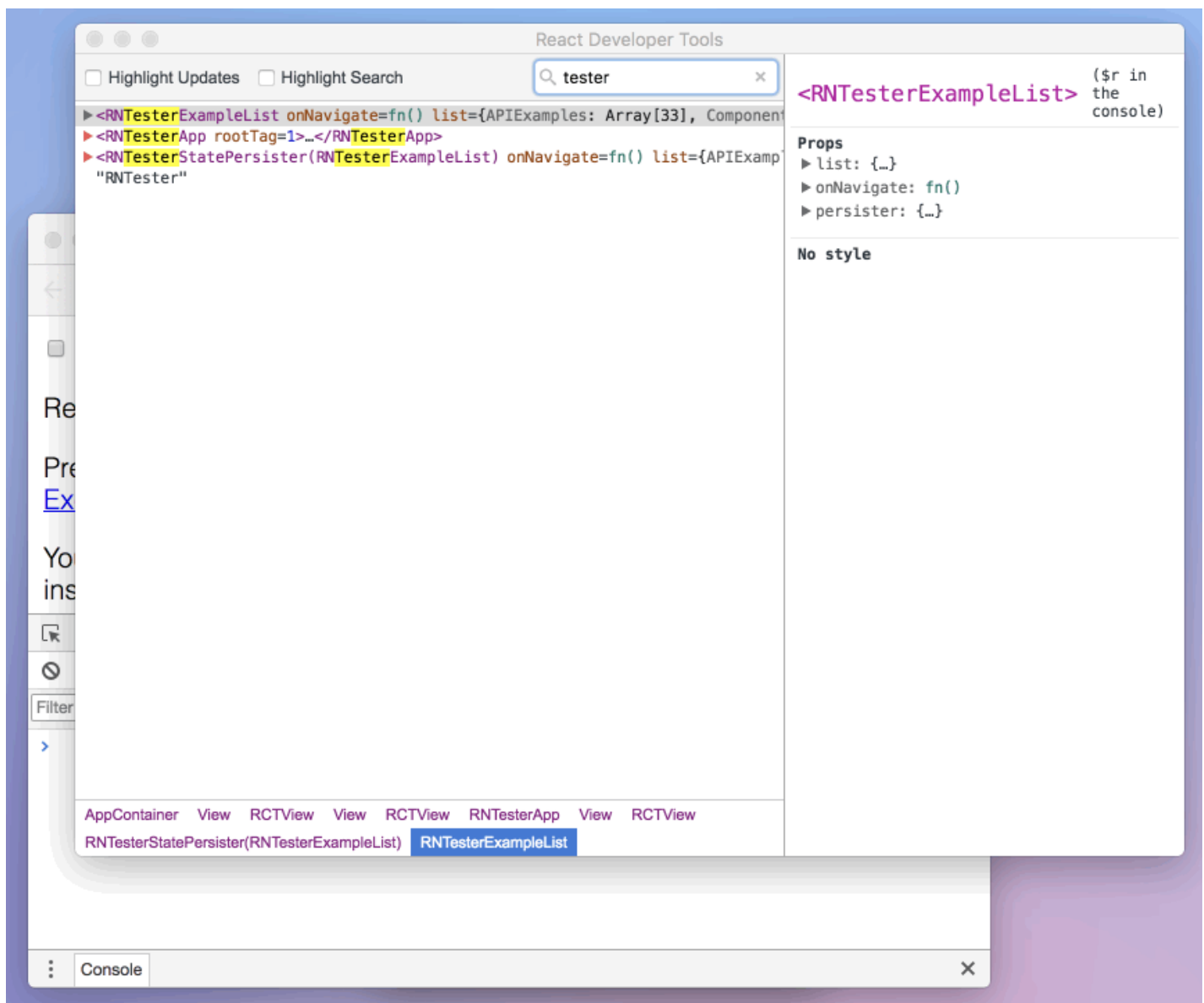
Inspecting Component Instances

When debugging JavaScript in Chrome, you can inspect the props and state of the React components in the browser console.

First, follow the instructions for debugging in Chrome to open the Chrome console.

Make sure that the dropdown in the top left corner of the Chrome console says `debuggerWorker.js`. **This step is essential.**

Then select a React component in React DevTools. There is a search box at the top that helps you find one by name. As soon as you select it, it will be available as `$r` in the Chrome console, letting you inspect its props, state, and instance properties.




Debugging application state

[Reactotron](#) is an open-source desktop app that allows you to inspect Redux or MobX-State-Tree application state as well as view custom logs, run custom commands such as resetting state, store and restore state snapshots, and other helpful debugging features for React Native apps.

You can view installation instructions [in the README](#). If you're using Expo, here is an [article detailing how to install on Expo](#).

Is this page useful?  

 Edit this page

*Last updated on **Jun 21, 2023***