MobileDrive

Debug React Native Apps

Because There Will Be Errors

Jack

2022.1.6

MobileDrive

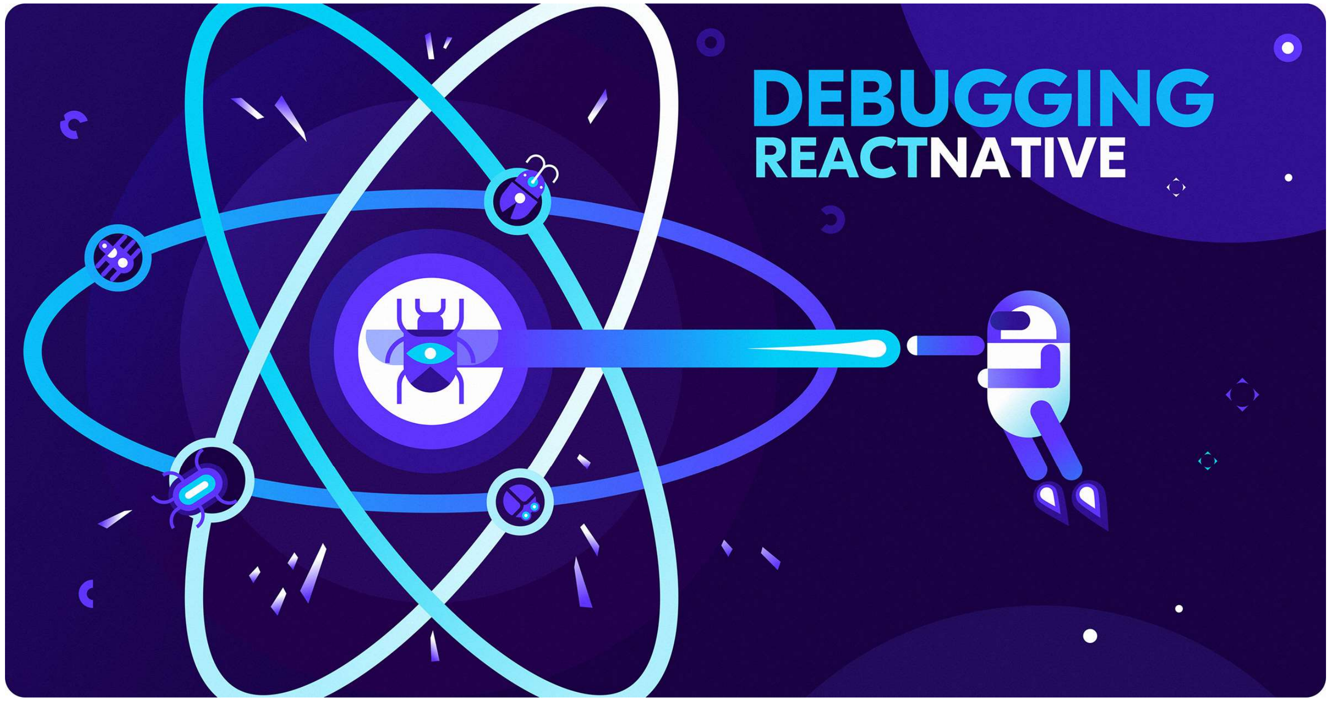
Table of Contents

Requirements

- A little React knowledge
- A little JavaScript knowledge

Introduction

- We will have a look at what we can do when things go wrong, namely at debugging React Native apps because there always will be some errors and there are different tools you can use to track down your errors, to fix them or to avoid them at runtime



What to Debug?

Error Messages / App Crashes	Logic Errors	Styling. Layout & UX
Syntax errors	Undesired or unexpected app behavior	Unexpected / “wrong” styling or layout
Bugs in your codes. (e.g. using undefined value, wrong types…)	Unexpected / Unhandled user behavior	Inconsistent result on different devices
“Unavoidable errors” (e.g. failing network requests)	Sequence of steps lead to errors	Layout doesn’t “work” on certain devices or orientations

MobileDrive

How to Debug?

Read the error messages (seriously)!

Often, you the error messages contains the solution or a (pretty) exact pointer at the problematic code line.

console.log()

Get a feeling for the “flow” of your code(What happens when? Which value is used where?)

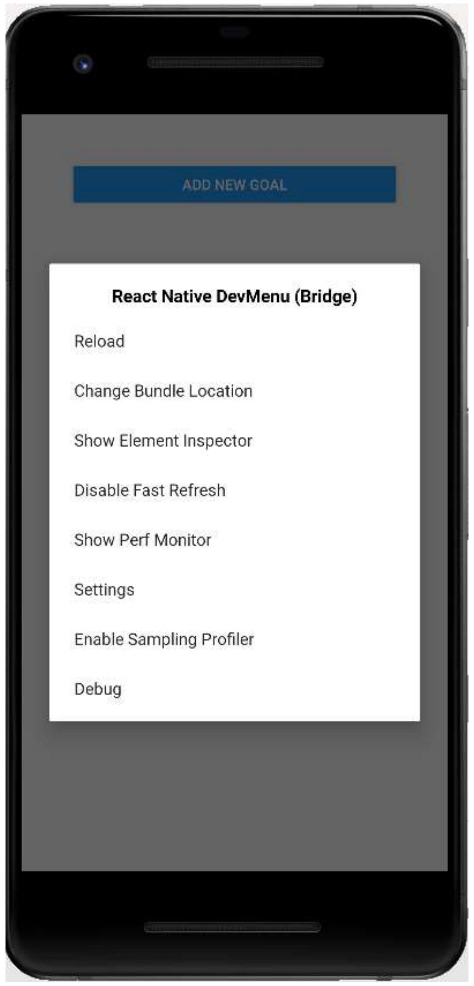
Other debuggers(e.g. Chrome)

Dive into the code in great detail and step by step.

Running the App on a Real Device & Debugging

- With the app running on a real device, you can debug it from there, too. **Shake the device a little** to bring up the developer menu.

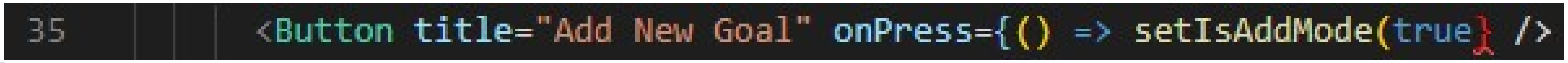
- There, you can enable the debugger and the other features covered on this slide.



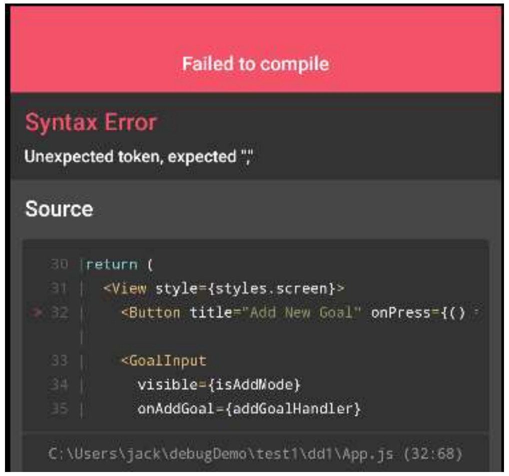
Handling Error Messages: Syntax Error

- Missing bracket

App.js

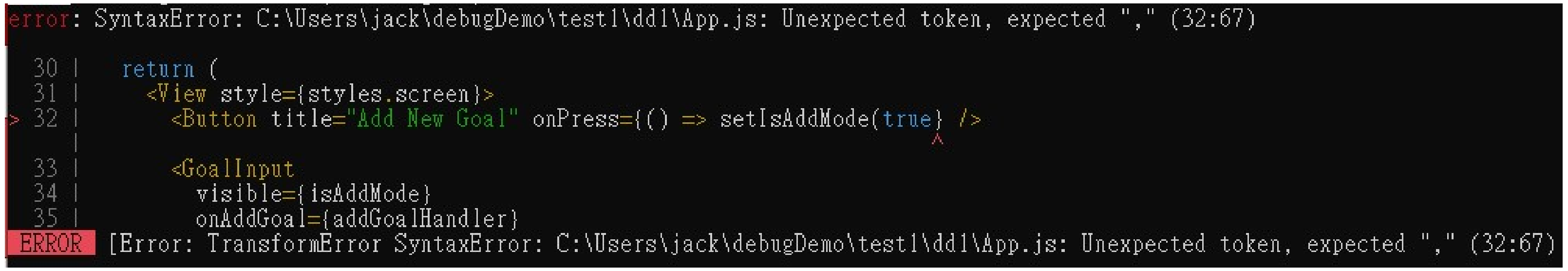


```
35      <Button title="Add New Goal" onPress={() => setIsAddMode(true} />
```



```
Failed to compile

Syntax Error
Unexpected token, expected ","

Source
  30 |return (
  31 |  <View style={styles.screen}>
> 32 |    <Button title="Add New Goal" onPress={() =
     |
  33 |    <GoalInput
  34 |      visible={isAddMode}
  35 |      onAddGoal={addGoalHandler}

C:\Users\jack\debugDemo\test1\dd1\App.js (32:68)
```

Device log



```
error: SyntaxError: C:\Users\jack\debugDemo\test1\dd1\App.js: Unexpected token, expected "," (32:67)

  30 |   return (
  31 |     <View style={styles.screen}>
> 32 |       <Button title="Add New Goal" onPress={() => setIsAddMode(true} />
     |                                                                   ^
  33 |       <GoalInput
  34 |         visible={isAddMode}
  35 |         onAddGoal={addGoalHandler}
 ERROR  [Error: TransformError SyntaxError: C:\Users\jack\debugDemo\test1\dd1\App.js: Unexpected token, expected "," (32:67)
```

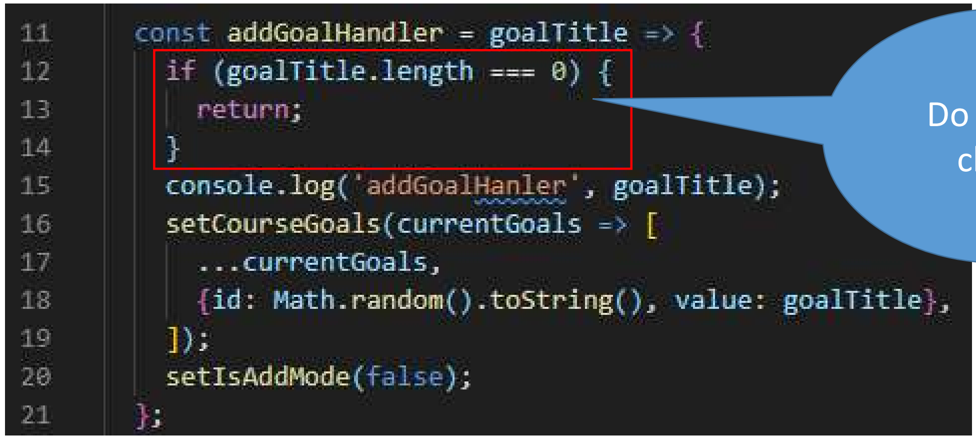
Metro log

It's still pointing us at the wrong solution, expecting a comma but it does point us at the right code

Handling Error Messages: Bugs in Your Codes

- Undefined value

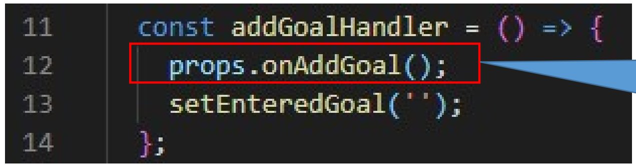
App.js



```
const addGoalHandler = goalTitle => {
  if (goalTitle.length === 0) {
    return;
  }
  console.log('addGoalHanler', goalTitle);
  setCourseGoals(currentGoals => [
    ...currentGoals,
    {id: Math.random().toString(), value: goalTitle},
  ]);
  setIsAddMode(false);
};
```

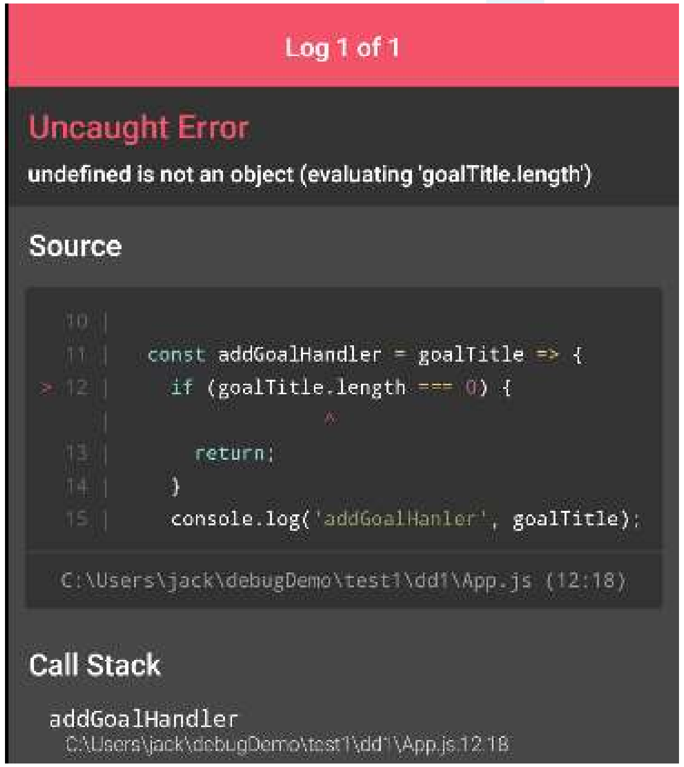
Do length check

GoalInput.js



```
const addGoalHandler = () => {
  props.onAddGoal();
  setEnteredGoal('');
};
```

Forgot to return value



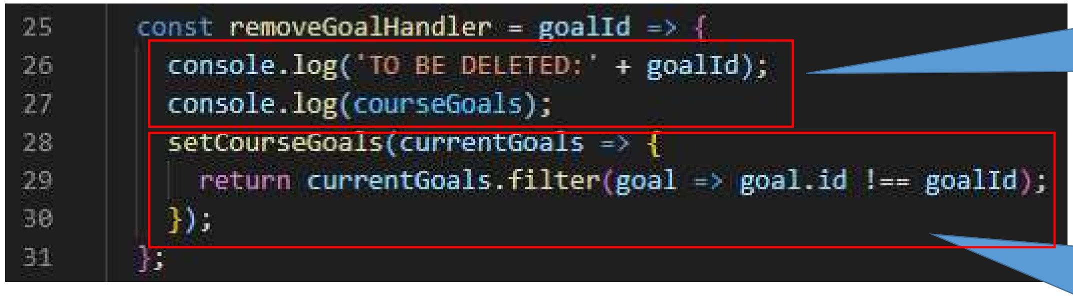
```
Log 1 of 1

Uncaught Error
undefined is not an object (evaluating 'goalTitle.length')

Source

   10 |
   11 |   const addGoalHandler = goalTitle => {
 > 12 |     if (goalTitle.length === 0) {
      |                   ^
   13 |       return;
   14 |     }
   15 |     console.log('addGoalHanler', goalTitle);

C:\Users\jack\debugDemo\test1\dd1\App.js (12:18)

Call Stack

addGoalHandler
  C:\Users\jack\debugDemo\test1\dd1\App.js:12:18
```

Device log

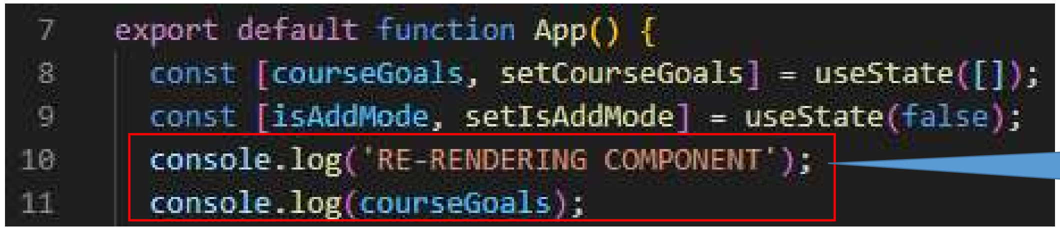
It informs us that the problem has something to do with something being undefined and that it is related to this goal title length check.

Understanding Code Flow with console.log()

App.js



```
const removeGoalHandler = goalId => {
  console.log('TO BE DELETED:' + goalId);
  console.log(courseGoals);
  setCourseGoals(currentGoals => {
    return currentGoals.filter(goal => goal.id !== goalId);
  });
};
```



```
export default function App() {
  const [courseGoals, setCourseGoals] = useState([]);
  const [isAddMode, setIsAddMode] = useState(false);
  console.log('RE-RENDERING COMPONENT');
  console.log(courseGoals);
```

Add some info text like to be deleted in front of this

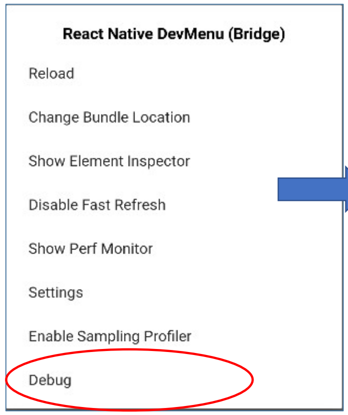
The entire component will be re-rendered when we update our course goals with this line

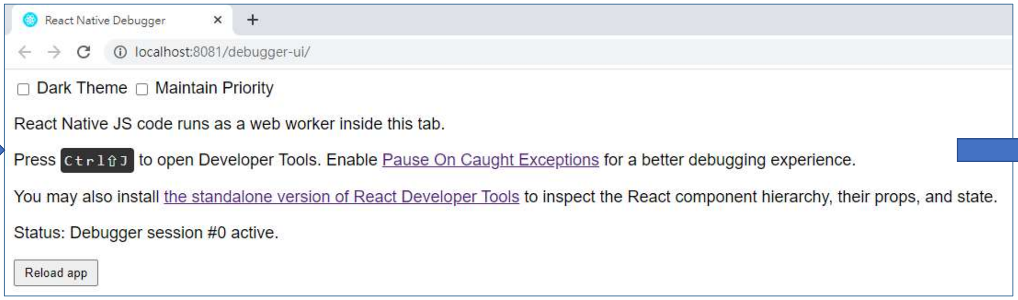
So we can maybe just also add a little console log, re-rendering component, so that we know that the course goal list we're printing after this one is the list after every re-render cycle

We of course see that that‘s behaving correctly here but simply imagine you have a more complex flow in your code you want to debug, then such console log statements can really help you understand how your code is running, how often it's running and if the correct values are getting used.

Using the Remote Debugger & Breakpoints: Launch

- Now sometimes, console log alone doesn't get you that far, you need more help and in such cases, you can debug your code remotely.
- On Android simulator, you press **control m** and it will open debug menu
- If you press **Debug**, a new tab should open up in the browser which automatically navigated to localhost http://localhost:8081/debugger-ui/
- You can open the Chrome developer tools now with the shortcut you should be seeing here



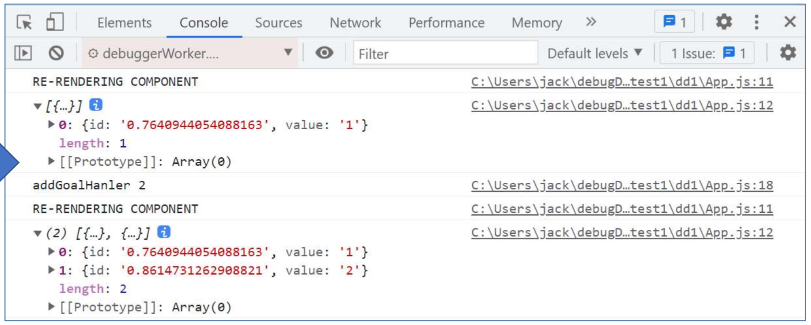


React Native JS code runs as a web worker inside this tab.

Press Ctrl⇧J to open Developer Tools. Enable Pause On Caught Exceptions for a better debugging experience.

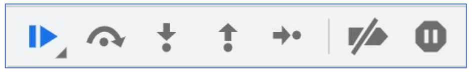
You may also install the standalone version of React Developer Tools to inspect the React component hierarchy, their props, and state.

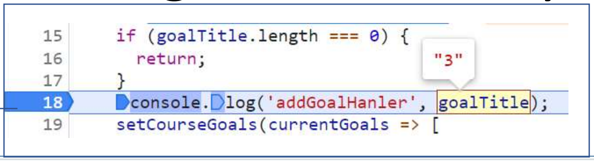
Status: Debugger session #0 active.



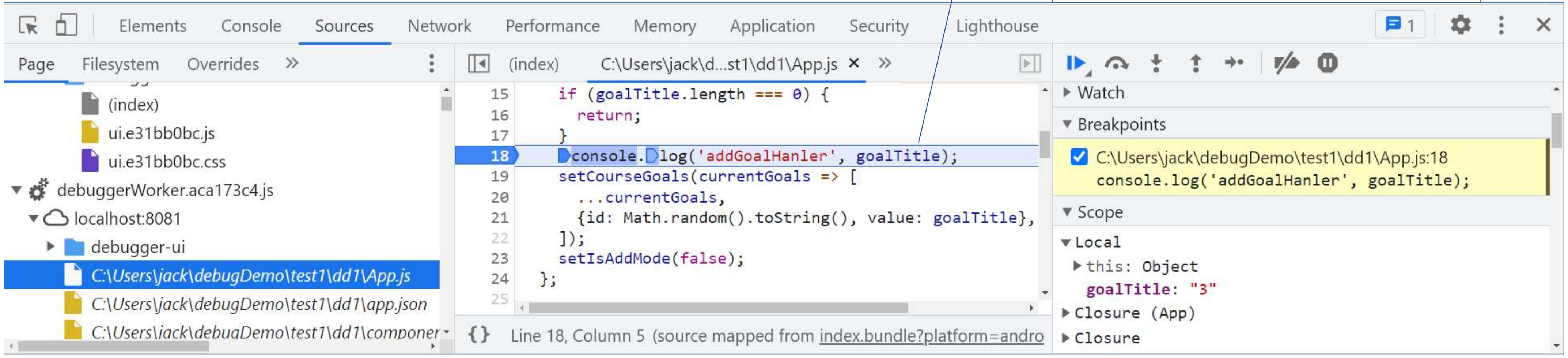
```
RE-RENDERING COMPONENT                    C:\Users\jack\debugD…test1\dd1\App.js:11
▼[{…}]                                    C:\Users\jack\debugD…test1\dd1\App.js:12
  ▶0: {id: '0.7640944054088163', value: '1'}
   length: 1
  ▶[[Prototype]]: Array(0)
addGoalHanler 2                           C:\Users\jack\debugD…test1\dd1\App.js:18
RE-RENDERING COMPONENT                    C:\Users\jack\debugD…test1\dd1\App.js:11
▼(2) [{…}, {…}]                           C:\Users\jack\debugD…test1\dd1\App.js:12
  ▶0: {id: '0.7640944054088163', value: '1'}
  ▶1: {id: '0.8614731262908821', value: '2'}
   length: 2
  ▶[[Prototype]]: Array(0)
```

Using the Remote Debugger & Breakpoints: Set Breakpoints

- You can dive into sources for example to dive into your source code and set breakpoints.
- Now the cool thing is you can walk through your code step-by-step with these controls  and you can also hover over things to look into your code, for example to see the current value in goal title.



```
if (goalTitle.length === 0) {
  return;
}
console.log('addGoalHanler', goalTitle);
setCourseGoals(currentGoals => [
```



Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Page
Filesystem
Overrides
(index)
ui.e31bb0bc.js
ui.e31bb0bc.css
debuggerWorker.aca173c4.js
localhost:8081
debugger-ui
C:\Users\jack\debugDemo\test1\dd1\App.js
C:\Users\jack\debugDemo\test1\dd1\app.json
C:\Users\jack\debugDemo\test1\dd1\componer
(index)
C:\Users\jack\d...st1\dd1\App.js
Line 18, Column 5 (source mapped from index.bundle?platform=andro
Watch
Breakpoints
Scope

```
  if (goalTitle.length === 0) {
    return;
  }
  console.log('addGoalHanler', goalTitle);
  setCourseGoals(currentGoals => [
    ...currentGoals,
    {id: Math.random().toString(), value: goalTitle},
  ]);
  setIsAddMode(false);
};
```

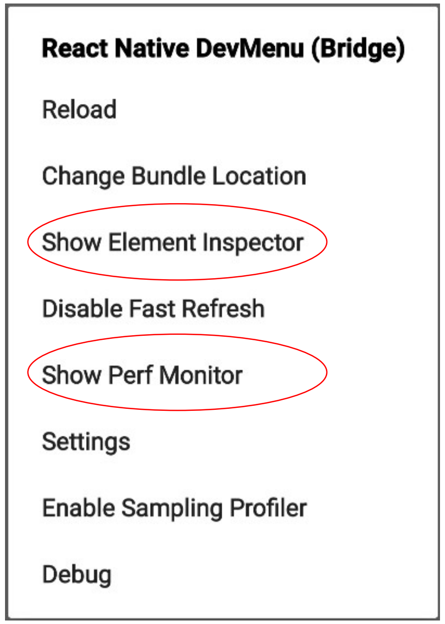
Breakpoints:
✓ C:\Users\jack\debugDemo\test1\dd1\App.js:18
`console.log('addGoalHanler', goalTitle);`

Scope:

```
▼ Local
  ▶ this: Object
    goalTitle: "3"
▶ Closure (App)
▶ Closure
```

Working with the Device DevTools Overlay

- If you enable **Perf Monitor**, you get this overlay which basically informs you about the performance you are having and there you can see at how many frames your app is running, how many frames were dropped and so on.
- If you enable **Element Inspector**, you can click onto items in your user interface to get information about them.



React Native DevMenu (Bridge)

Reload

Change Bundle Location

Show Element Inspector

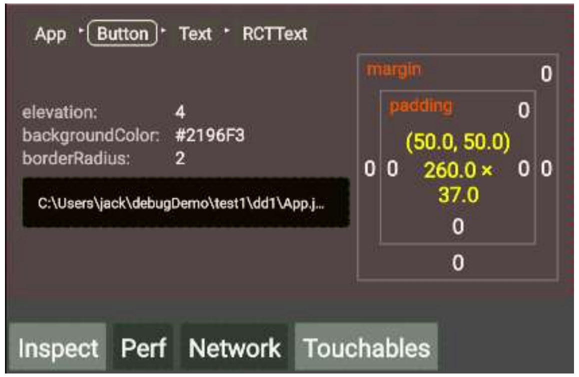
Disable Fast Refresh

Show Perf Monitor

Settings

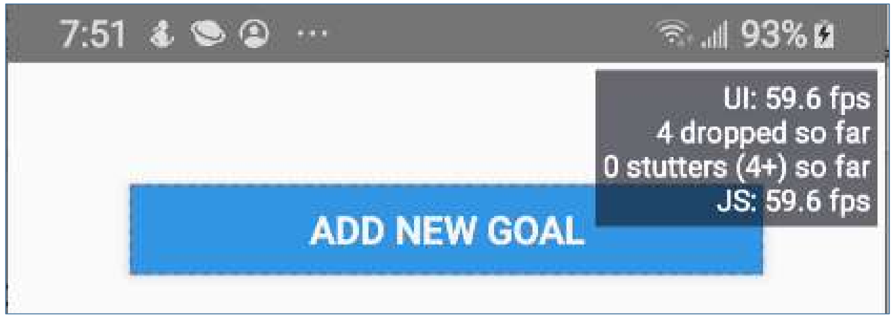
Enable Sampling Profiler

Debug



```
elevation:        4
backgroundColor:  #2196F3
borderRadius:     2
C:\Users\jack\debugDemo\test1\dd1\App.j...
```

```
margin 0
padding 0
(50.0, 50.0)
0 0  260.0 × 37.0  0 0
0
0
```

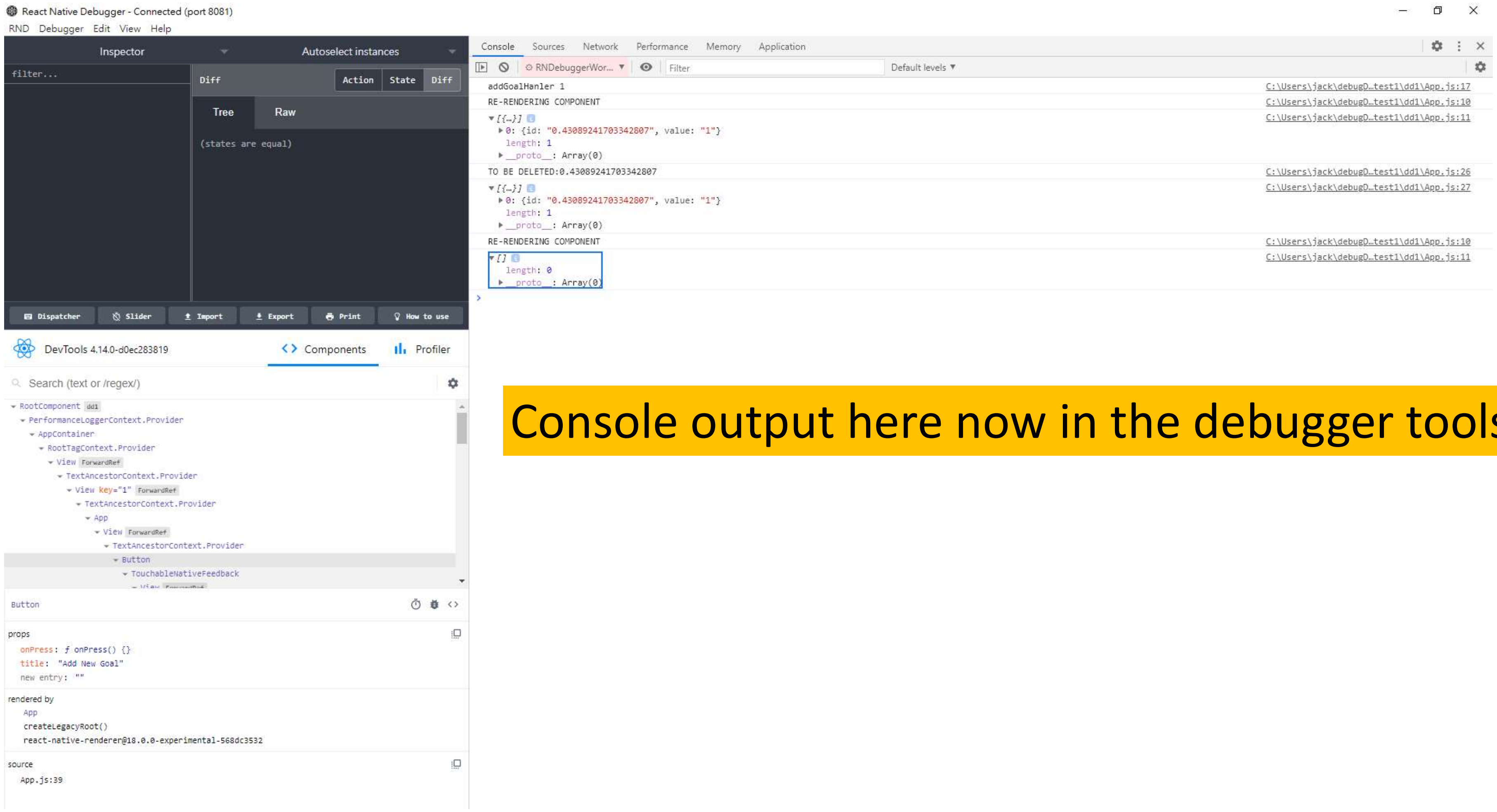


```
7:51                          93%
UI: 59.6 fps
4 dropped so far
0 stutters (4+) so far
JS: 59.6 fps
```

Debugging the UI & Using React Native Debugger

- There are several even better tools for inspecting the user interface, like **React Native Debugger**
- For this to work, enable remote JavaScript debugging on the devices, just what we did before to debug this in Chrome.
- Now with this opened up, press command t on Mac or **control t** on Windows or Linux in here to open a new tab and open and confirm that React Native debugger port which the Chrome tab also used before and confirm this and now it's trying to connect there and to make this succeed, and you'll see your console output here now in the debugger tools.

React Native Debugger: Console Output



```
(states are equal)
```

```
addGoalHanler 1                                          C:\Users\jack\debugD…test1\dd1\App.js:17
RE-RENDERING COMPONENT                                   C:\Users\jack\debugD…test1\dd1\App.js:10
▼[{…}]                                                   C:\Users\jack\debugD…test1\dd1\App.js:11
  ▶0: {id: "0.43089241703342807", value: "1"}
   length: 1
  ▶__proto__: Array(0)
TO BE DELETED:0.43089241703342807                        C:\Users\jack\debugD…test1\dd1\App.js:26
▼[{…}]                                                   C:\Users\jack\debugD…test1\dd1\App.js:27
  ▶0: {id: "0.43089241703342807", value: "1"}
   length: 1
  ▶__proto__: Array(0)
RE-RENDERING COMPONENT                                   C:\Users\jack\debugD…test1\dd1\App.js:10
▼[]                                                      C:\Users\jack\debugD…test1\dd1\App.js:11
   length: 0
  ▶__proto__: Array(0)
>
```

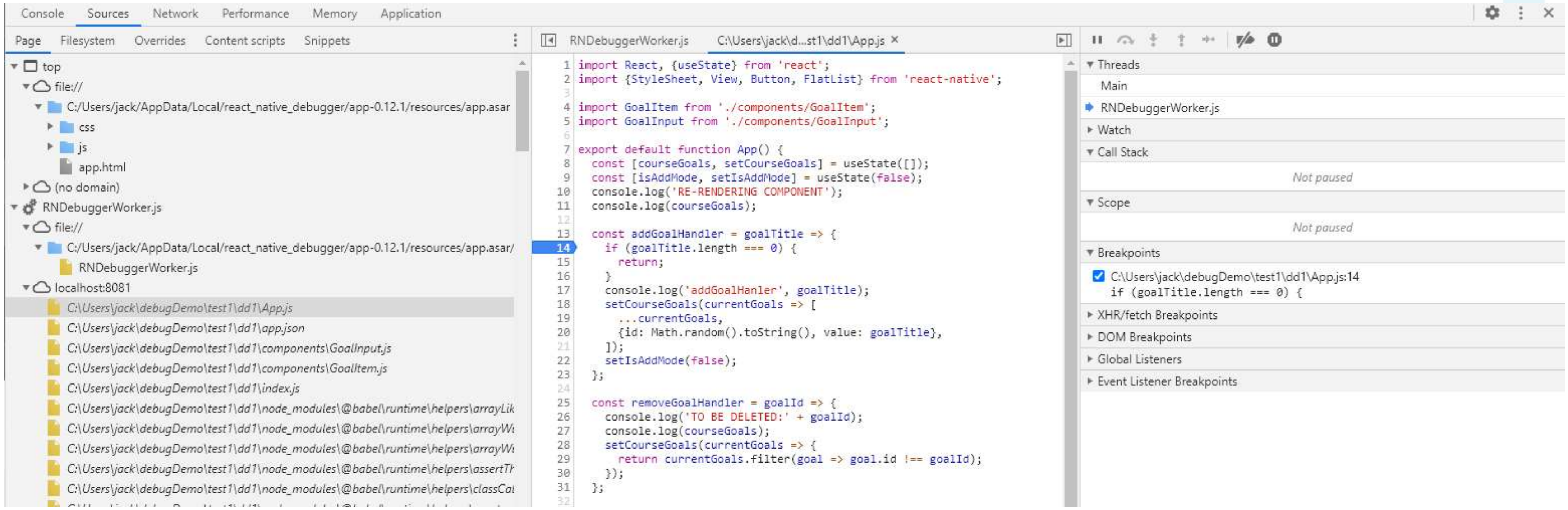
```
▾ RootComponent dd1
  ▾ PerformanceLoggerContext.Provider
    ▾ AppContainer
      ▾ RootTagContext.Provider
        ▾ View ForwardRef
          ▾ TextAncestorContext.Provider
            ▾ View key="1" ForwardRef
              ▾ TextAncestorContext.Provider
                ▾ App
                  ▾ View ForwardRef
                    ▾ TextAncestorContext.Provider
                      ▾ Button
                        ▾ TouchableNativeFeedback
```

```
Button

props
  onPress: ƒ onPress() {}
  title: "Add New Goal"
  new entry: ""

rendered by
  App
  createLegacyRoot()
  react-native-renderer@18.0.0-experimental-568dc3532

source
  App.js:39
```

Console output here now in the debugger tools

MobileDrive

React Native Debugger: Sources



```
top
  file://
    C:/Users/jack/AppData/Local/react_native_debugger/app-0.12.1/resources/app.asar
      css
      js
      app.html
  (no domain)
RNDebuggerWorker.js
  file://
    C:/Users/jack/AppData/Local/react_native_debugger/app-0.12.1/resources/app.asar/
      RNDebuggerWorker.js
  localhost:8081
    C:\Users\jack\debugDemo\test1\dd1\App.js
    C:\Users\jack\debugDemo\test1\dd1\app.json
    C:\Users\jack\debugDemo\test1\dd1\components\GoalInput.js
    C:\Users\jack\debugDemo\test1\dd1\components\GoalItem.js
    C:\Users\jack\debugDemo\test1\dd1\index.js
    C:\Users\jack\debugDemo\test1\dd1\node_modules\@babel\runtime\helpers\arrayLik
    C:\Users\jack\debugDemo\test1\dd1\node_modules\@babel\runtime\helpers\arrayWi
    C:\Users\jack\debugDemo\test1\dd1\node_modules\@babel\runtime\helpers\arrayWi
    C:\Users\jack\debugDemo\test1\dd1\node_modules\@babel\runtime\helpers\assertTh
    C:\Users\jack\debugDemo\test1\dd1\node_modules\@babel\runtime\helpers\classCal
```

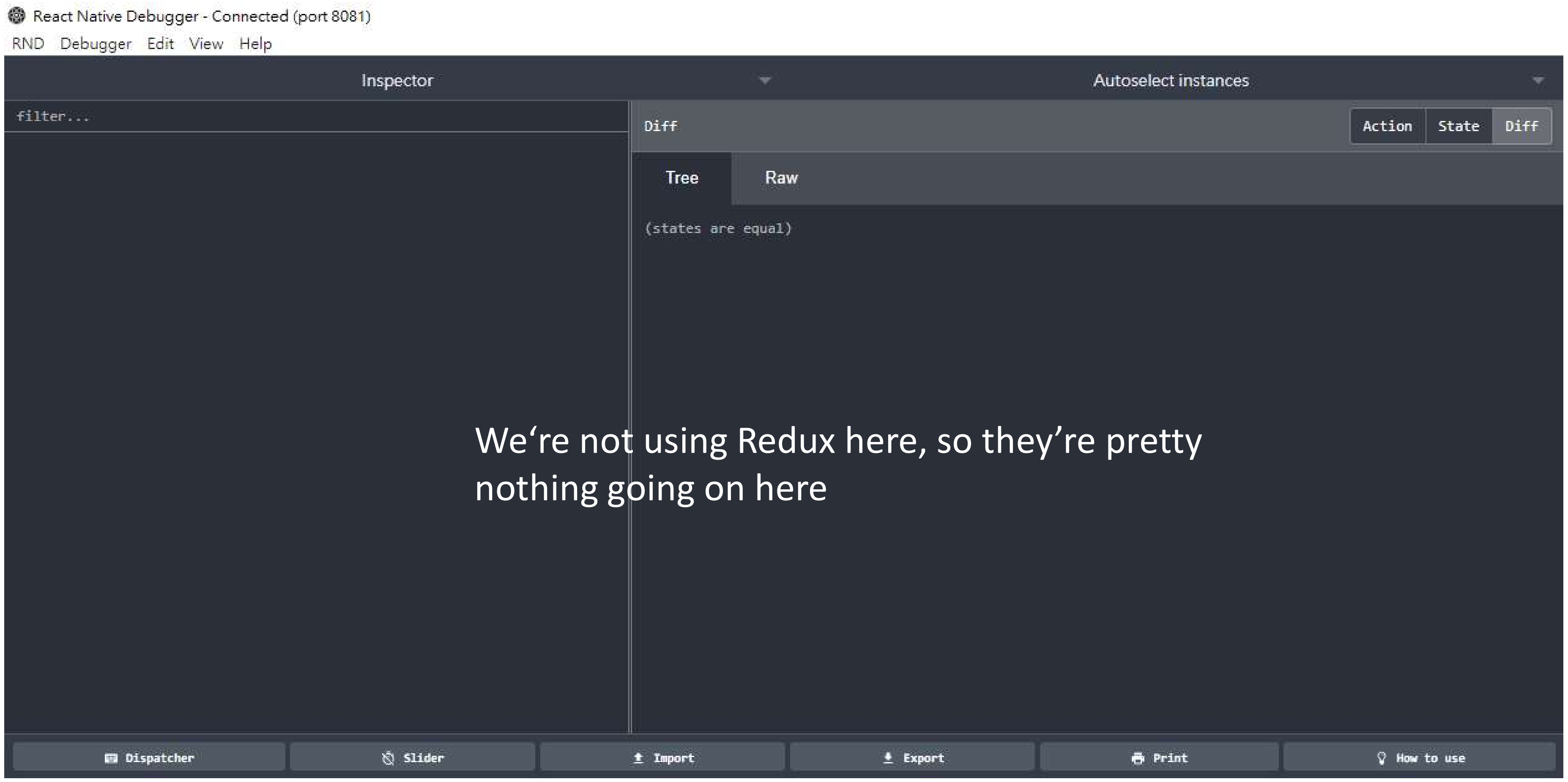
```javascript
import React, {useState} from 'react';
import {StyleSheet, View, Button, FlatList} from 'react-native';

import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';

export default function App() {
  const [courseGoals, setCourseGoals] = useState([]);
  const [isAddMode, setIsAddMode] = useState(false);
  console.log('RE-RENDERING COMPONENT');
  console.log(courseGoals);

  const addGoalHandler = goalTitle => {
    if (goalTitle.length === 0) {
      return;
    }
    console.log('addGoalHanler', goalTitle);
    setCourseGoals(currentGoals => [
      ...currentGoals,
      {id: Math.random().toString(), value: goalTitle},
    ]);
    setIsAddMode(false);
  };

  const removeGoalHandler = goalId => {
    console.log('TO BE DELETED:' + goalId);
    console.log(courseGoals);
    setCourseGoals(currentGoals => {
      return currentGoals.filter(goal => goal.id !== goalId);
    });
  };
```

```
▼ Threads
  Main
  RNDebuggerWorker.js
▶ Watch
▼ Call Stack
  Not paused
▼ Scope
  Not paused
▼ Breakpoints
  ☑ C:\Users\jack\debugDemo\test1\dd1\App.js:14
    if (goalTitle.length === 0) {
▶ XHR/fetch Breakpoints
▶ DOM Breakpoints
▶ Global Listeners
▶ Event Listener Breakpoints
```

You'll also see, if I expand this, that in sources, you can again dive into your code here if you want to.

FIH CONFIDENTIAL

React Native Debugger: Redux

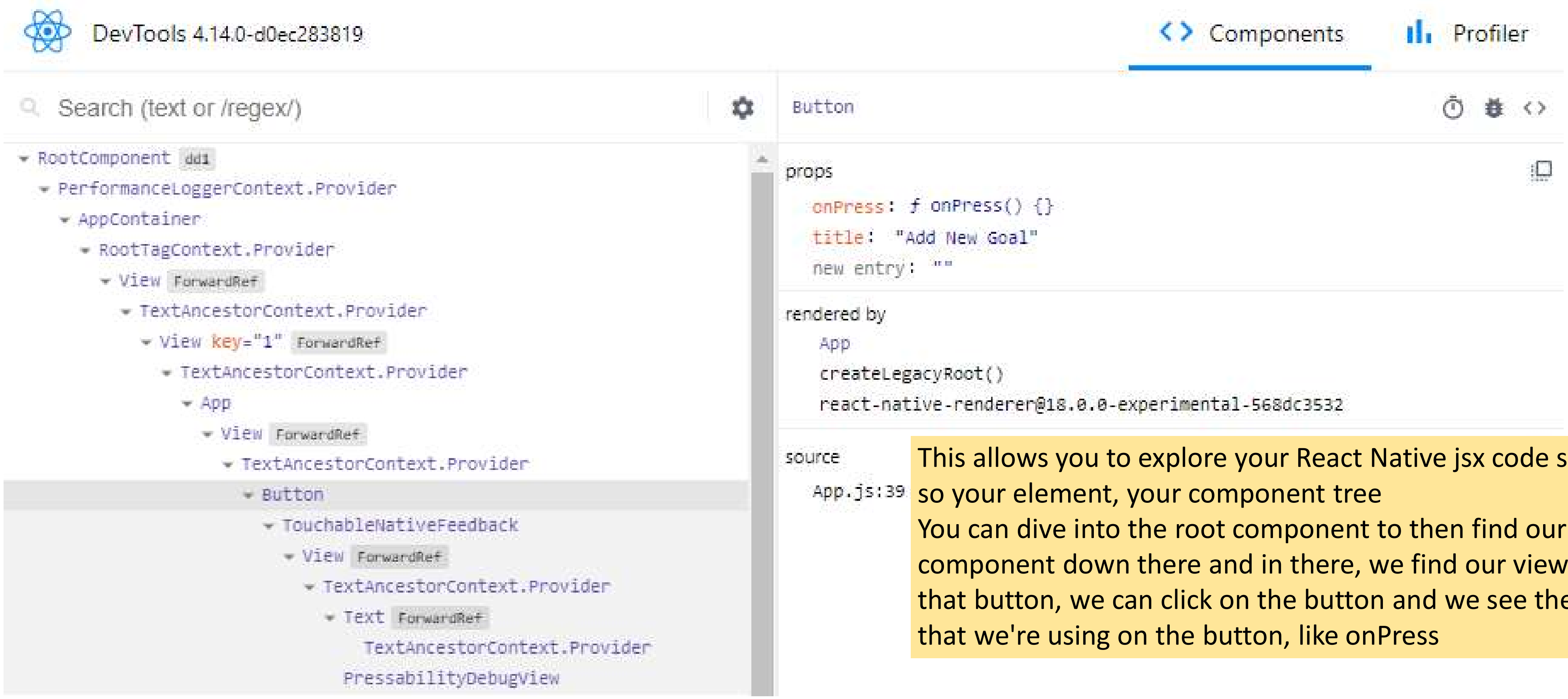


```
(states are equal)
```

We're not using Redux here, so they're pretty nothing going on here

Redux debugging tools

React Native Debugger: Elements



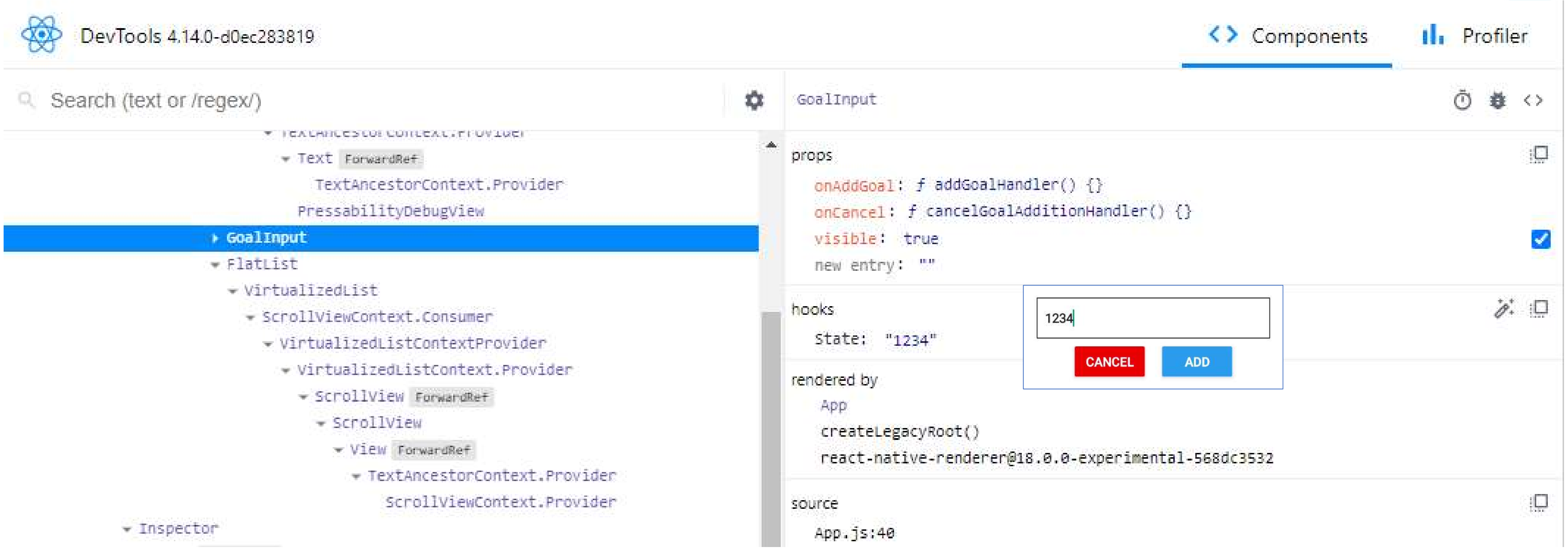
```
▾ RootComponent  dd1
  ▾ PerformanceLoggerContext.Provider
    ▾ AppContainer
      ▾ RootTagContext.Provider
        ▾ View  ForwardRef
          ▾ TextAncestorContext.Provider
            ▾ View key="1"  ForwardRef
              ▾ TextAncestorContext.Provider
                ▾ App
                  ▾ View  ForwardRef
                    ▾ TextAncestorContext.Provider
                      ▾ Button
                        ▾ TouchableNativeFeedback
                          ▾ View  ForwardRef
                            ▾ TextAncestorContext.Provider
                              ▾ Text  ForwardRef
                                  TextAncestorContext.Provider
                                PressabilityDebugView
```

```
Button

props
  onPress: ƒ onPress() {}
  title: "Add New Goal"
  new entry: ""

rendered by
  App
  createLegacyRoot()
  react-native-renderer@18.0.0-experimental-568dc3532

source
  App.js:39
```

This allows you to explore your React Native jsx code so to say, so your element, your component tree
You can dive into the root component to then find our app component down there and in there, we find our view with that button, we can click on the button and we see the props that we're using on the button, like onPress

Elements debugging tool

React Native Debugger: Props and States of Elements



```
Text ForwardRef
  TextAncestorContext.Provider
PressabilityDebugView
GoalInput
FlatList
  VirtualizedList
    ScrollViewContext.Consumer
      VirtualizedListContextProvider
        VirtualizedListContext.Provider
          ScrollView ForwardRef
            ScrollView
              View ForwardRef
                TextAncestorContext.Provider
                  ScrollViewContext.Provider
Inspector
```

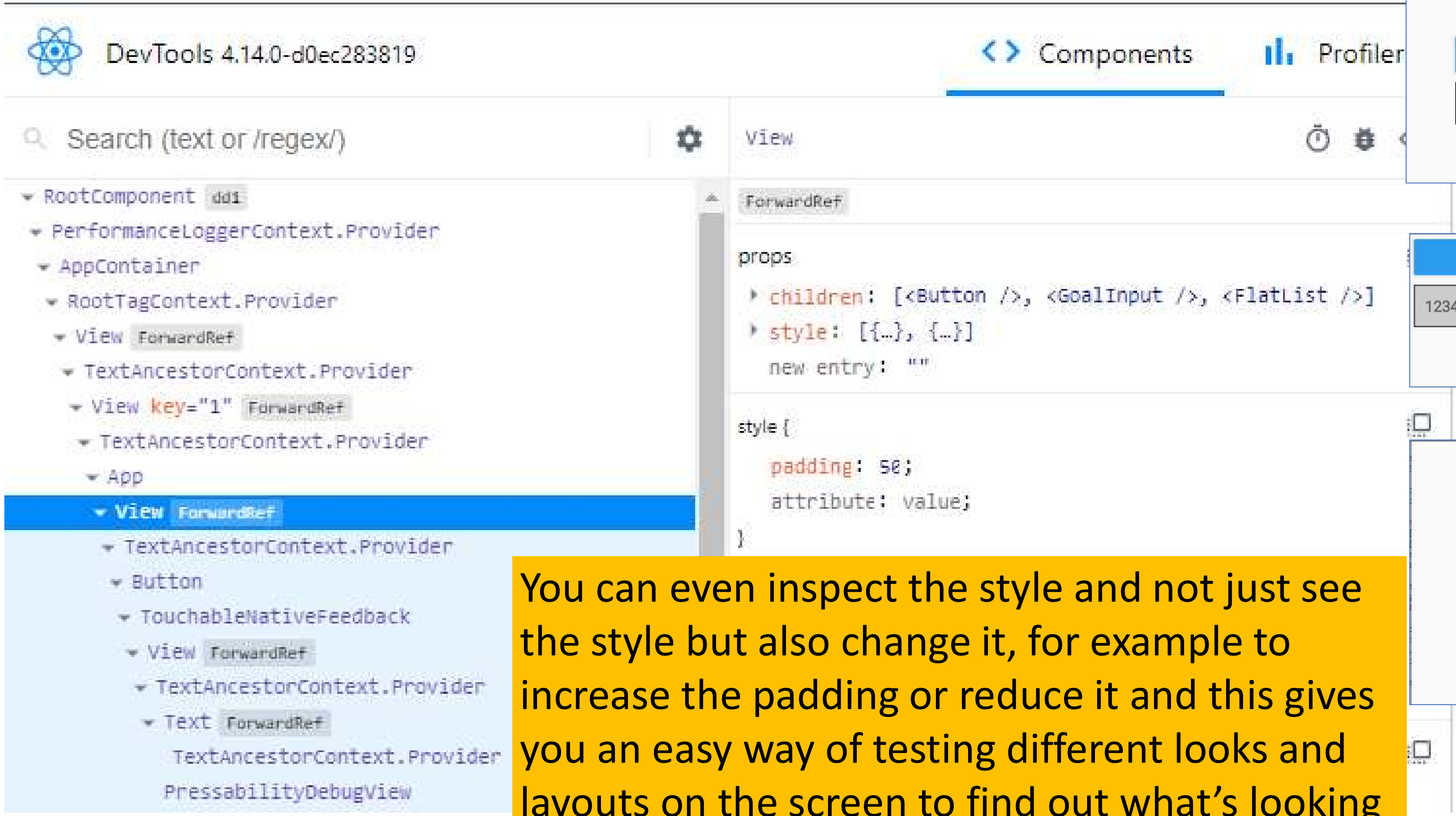
```
GoalInput

props
  onAddGoal: ƒ addGoalHandler() {}
  onCancel: ƒ cancelGoalAdditionHandler() {}
  visible: true
  new entry: ""

hooks
  State: "1234"

rendered by
  App
  createLegacyRoot()
  react-native-renderer@18.0.0-experimental-568dc3532

source
  App.js:40
```

1234

We can even change the visible prop to toggle this modal like this if we want to

if I start typing here, like 1234, you'll see that this also updates here

React Native Debugger: Styles



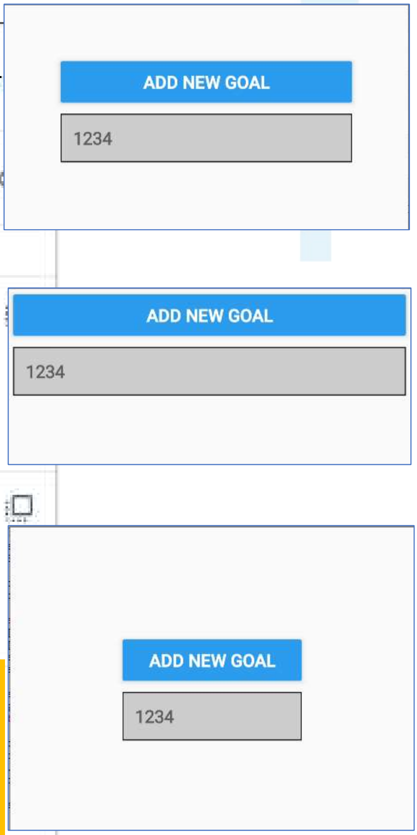
DevTools 4.14.0-d0ec283819

Components Profiler

Search (text or /regex/)

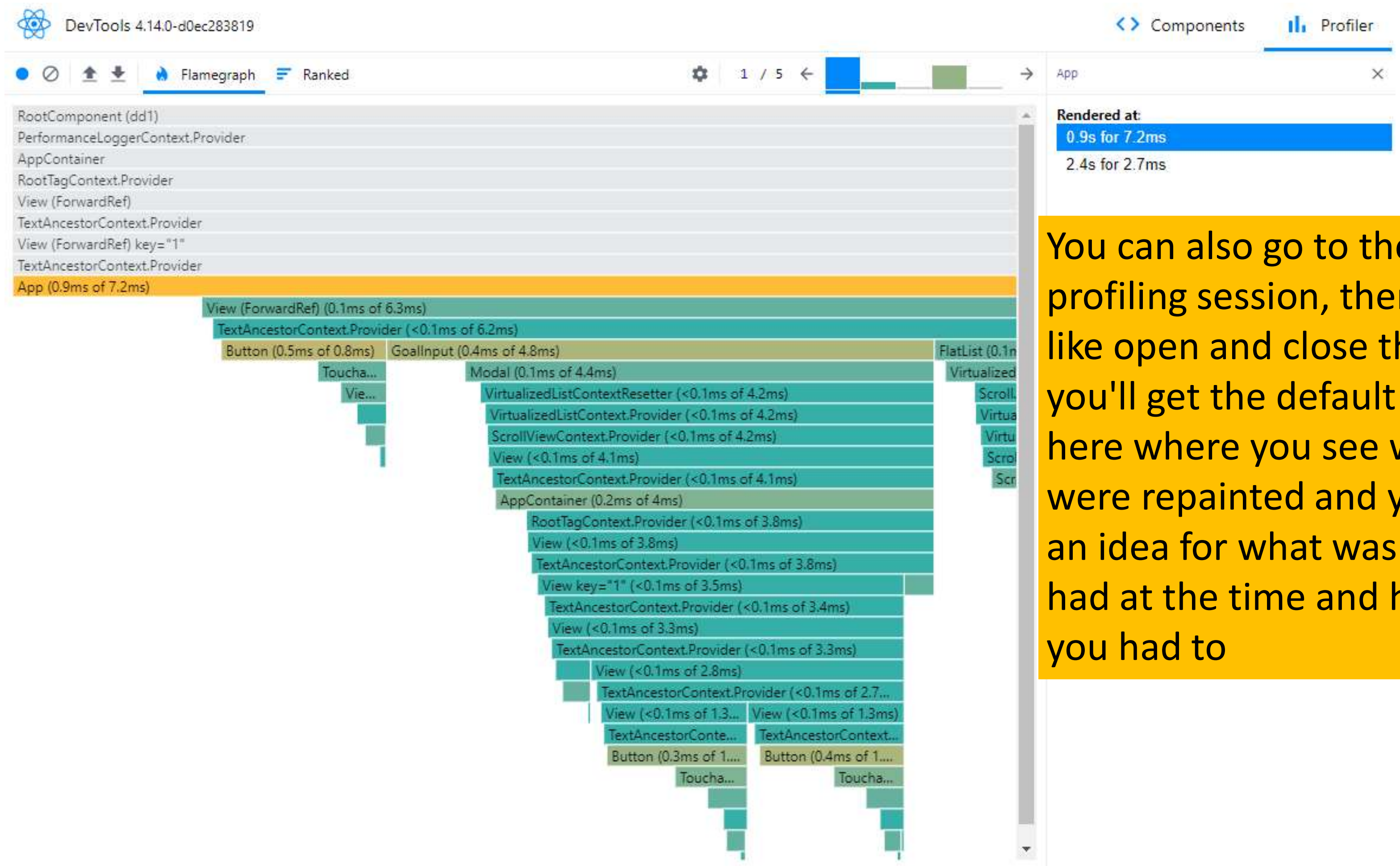
```
▾ RootComponent dd1
 ▾ PerformanceLoggerContext.Provider
  ▾ AppContainer
   ▾ RootTagContext.Provider
    ▾ View ForwardRef
     ▾ TextAncestorContext.Provider
      ▾ View key="1" ForwardRef
       ▾ TextAncestorContext.Provider
        ▾ App
         ▾ View ForwardRef
          ▾ TextAncestorContext.Provider
           ▾ Button
            ▾ TouchableNativeFeedback
             ▾ View ForwardRef
              ▾ TextAncestorContext.Provider
               ▾ Text ForwardRef
                  TextAncestorContext.Provider
                 PressabilityDebugView
```

```
View

ForwardRef

props
 ▸ children: [<Button />, <GoalInput />, <FlatList />]
 ▸ style: [{…}, {…}]
   new entry: ""

style {
   padding: 50;
   attribute: value;
}
```



You can even inspect the style and not just see the style but also change it, for example to increase the padding or reduce it and this gives you an easy way of testing different looks and layouts on the screen to find out what's looking good for you and what you want to change.

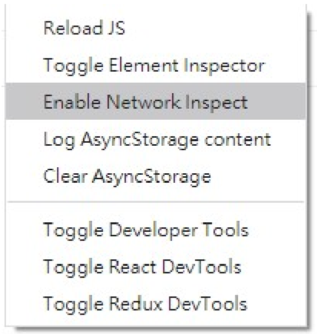
React Native Debugger: Profiler

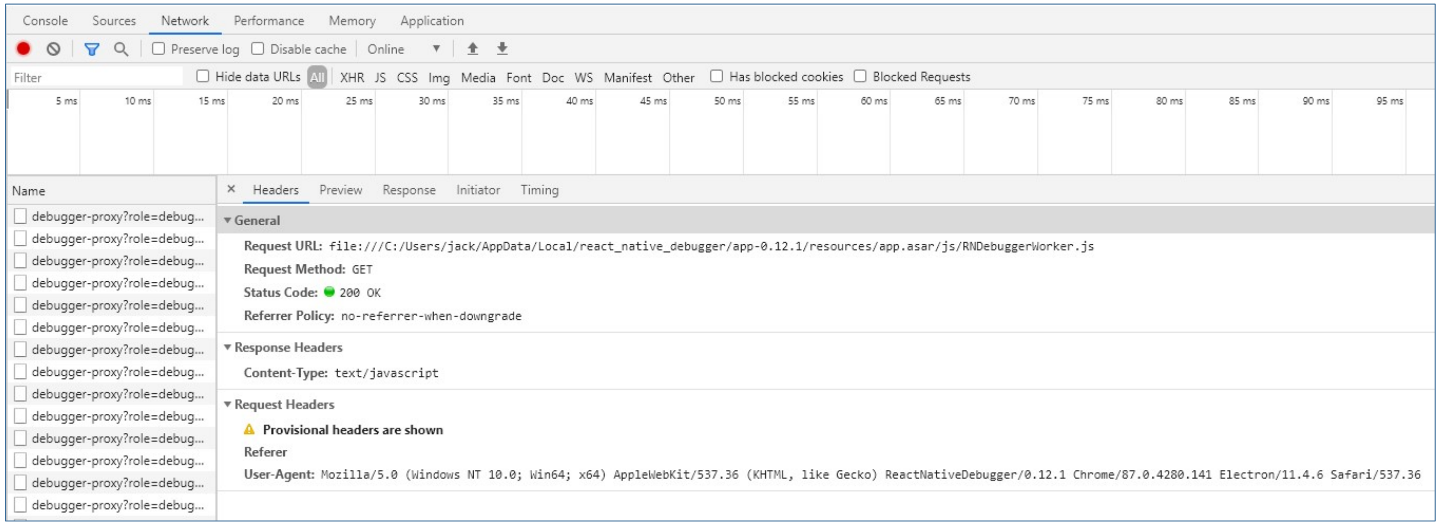

DevTools 4.14.0-d0ec283819
Components
Profiler
Flamegraph
Ranked
1 / 5
App
Rendered at:
0.9s for 7.2ms
2.4s for 2.7ms
RootComponent (dd1)
PerformanceLoggerContext.Provider
AppContainer
RootTagContext.Provider
View (ForwardRef)
TextAncestorContext.Provider
View (ForwardRef) key="1"
TextAncestorContext.Provider
App (0.9ms of 7.2ms)
View (ForwardRef) (0.1ms of 6.3ms)
TextAncestorContext.Provider (<0.1ms of 6.2ms)
Button (0.5ms of 0.8ms)
GoalInput (0.4ms of 4.8ms)
FlatList (0.1m
Toucha...
Vie...
Modal (0.1ms of 4.4ms)
Virtualized
VirtualizedListContextResetter (<0.1ms of 4.2ms)
VirtualizedListContext.Provider (<0.1ms of 4.2ms)
ScrollViewContext.Provider (<0.1ms of 4.2ms)
View (<0.1ms of 4.1ms)
TextAncestorContext.Provider (<0.1ms of 4.1ms)
AppContainer (0.2ms of 4ms)
RootTagContext.Provider (<0.1ms of 3.8ms)
View (<0.1ms of 3.8ms)
TextAncestorContext.Provider (<0.1ms of 3.8ms)
View key="1" (<0.1ms of 3.5ms)
TextAncestorContext.Provider (<0.1ms of 3.4ms)
View (<0.1ms of 3.3ms)
TextAncestorContext.Provider (<0.1ms of 3.3ms)
View (<0.1ms of 2.8ms)
TextAncestorContext.Provider (<0.1ms of 2.7...
View (<0.1ms of 1.3...
View (<0.1ms of 1.3ms)
TextAncestorConte...
TextAncestorContext...
Button (0.3ms of 1....
Button (0.4ms of 1....
Toucha...
Toucha...

You can also go to the profiler tab here and start a profiling session, then do something on the screen like open and close the modal and stop this and you'll get the default React dev tools experience here where you see which of your components were repainted and you can dive into that to get an idea for what was repainted, which props it had at the time and how many re-render cycles you had to

React Native Debugger: Network

- You can right click anywhere here, , then you can go to the network tab and you'll see outgoing network requests.
- Now these are all just debugging related requests but later in the course when we'll add our own network requests, where we send requests to our own web server, we can even inspect those here and look into them and see if we're sending and receiving the right data, something which is otherwise pretty hard to do with.



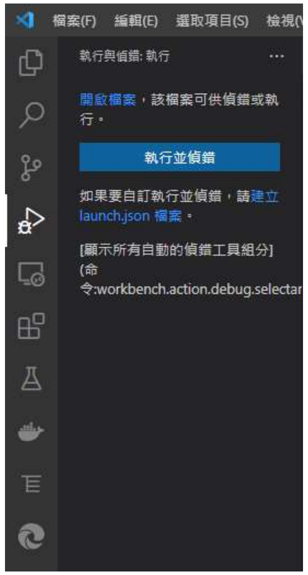


```
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
debugger-proxy?role=debug...
```

```
▼ General
  Request URL: file:///C:/Users/jack/AppData/Local/react_native_debugger/app-0.12.1/resources/app.asar/js/RNDebuggerWorker.js
  Request Method: GET
  Status Code: ● 200 OK
  Referrer Policy: no-referrer-when-downgrade
▼ Response Headers
  Content-Type: text/javascript
▼ Request Headers
  ⚠ Provisional headers are shown
  Referer
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) ReactNativeDebugger/0.12.1 Chrome/87.0.4280.141 Electron/11.4.6 Safari/537.36
```

Debugging React Native in VS Code: Prerequisites

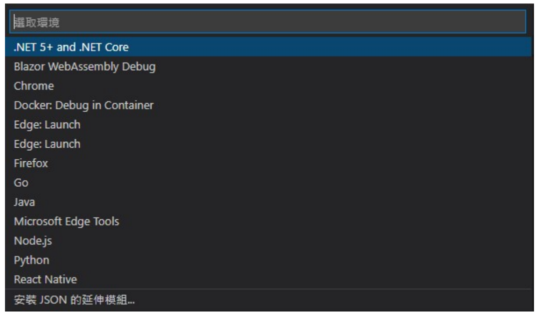
- VS Code installed
- React Native environment set up: https://reactnative.dev/docs/environment-setup
- Install Microsoft's React Native Tools extension for VS Code
- Open launch.json from the .vscode directory and add debug configurations

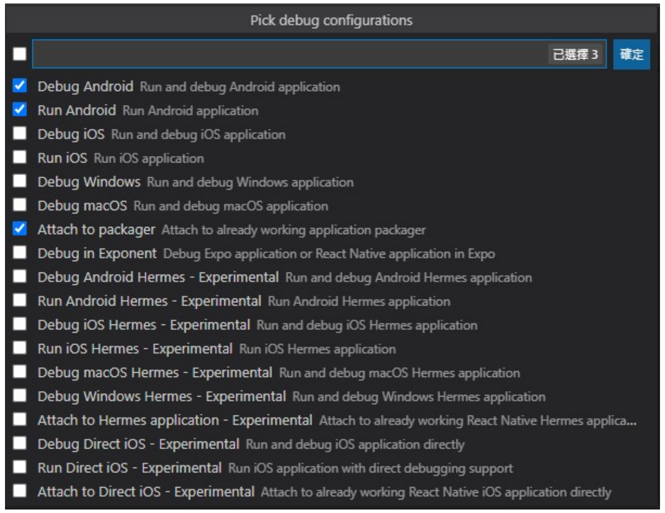


開啟檔案，該檔案可供偵錯或執行。

如果要自訂執行並偵錯，請建立 launch.json 檔案。

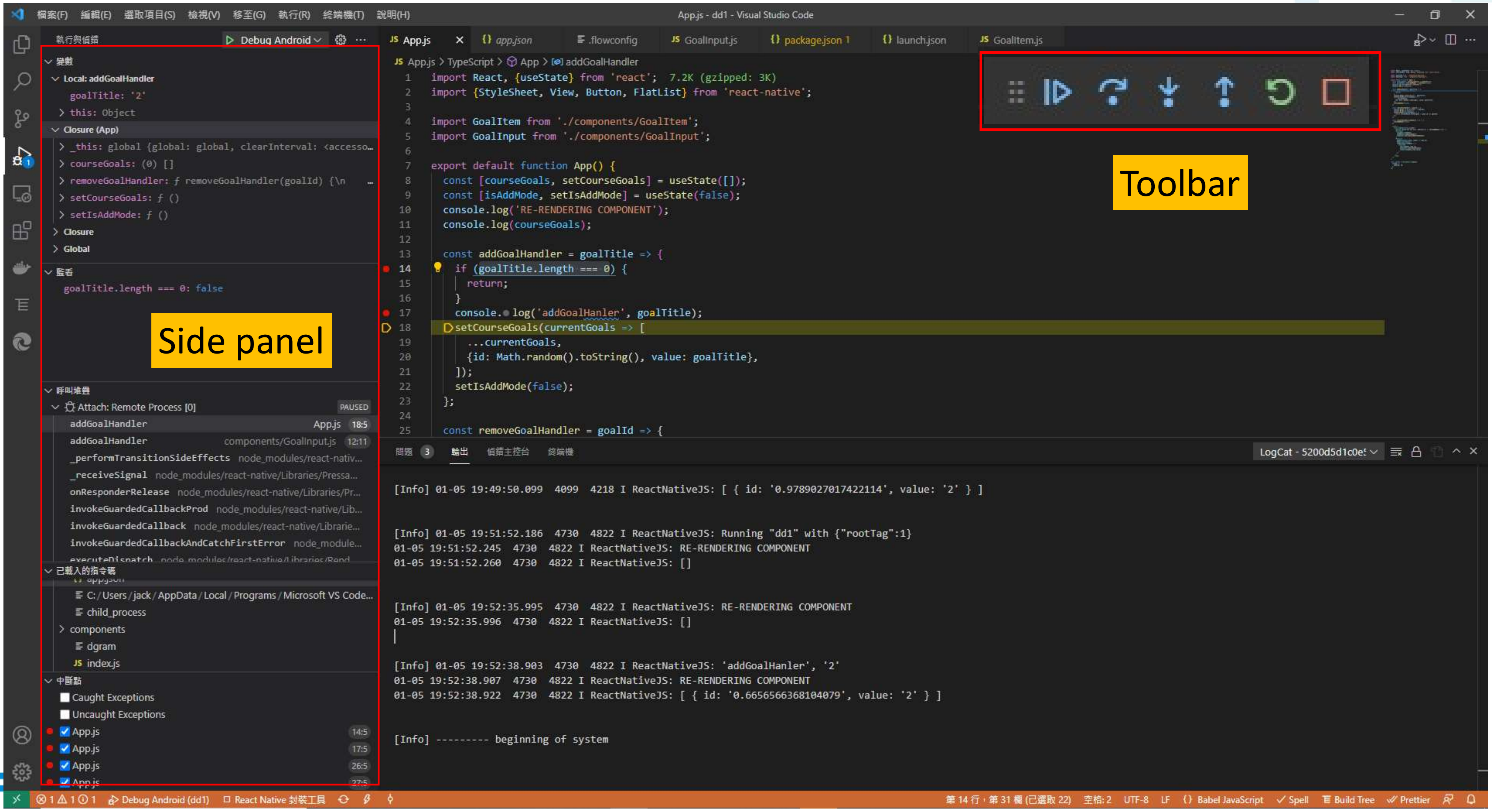
[顯示所有自動的偵錯工具組分] (命令:workbench.action.debug.selectar





- ✓ Debug Android Run and debug Android application
- ✓ Run Android Run Android application
- Debug iOS Run and debug iOS application
- Run iOS Run iOS application
- Debug Windows Run and debug Windows application
- Debug macOS Run and debug macOS application
- ✓ Attach to packager Attach to already working application packager
- Debug in Exponent Debug Expo application or React Native application in Expo
- Debug Android Hermes - Experimental Run and debug Android Hermes application
- Run Android Hermes - Experimental Run Android Hermes application
- Debug iOS Hermes - Experimental Run and debug iOS Hermes application
- Run iOS Hermes - Experimental Run iOS Hermes application
- Debug macOS Hermes - Experimental Run and debug macOS Hermes application
- Debug Windows Hermes - Experimental Run and debug Windows Hermes application
- Attach to Hermes application - Experimental Attach to already working React Native Hermes applica...
- Debug Direct iOS - Experimental Run and debug iOS application directly
- Run Direct iOS - Experimental Run iOS application with direct debugging support
- Attach to Direct iOS - Experimental Attach to already working React Native iOS application directly

Debugging React Native in VS Code: Enjoy Debugging!

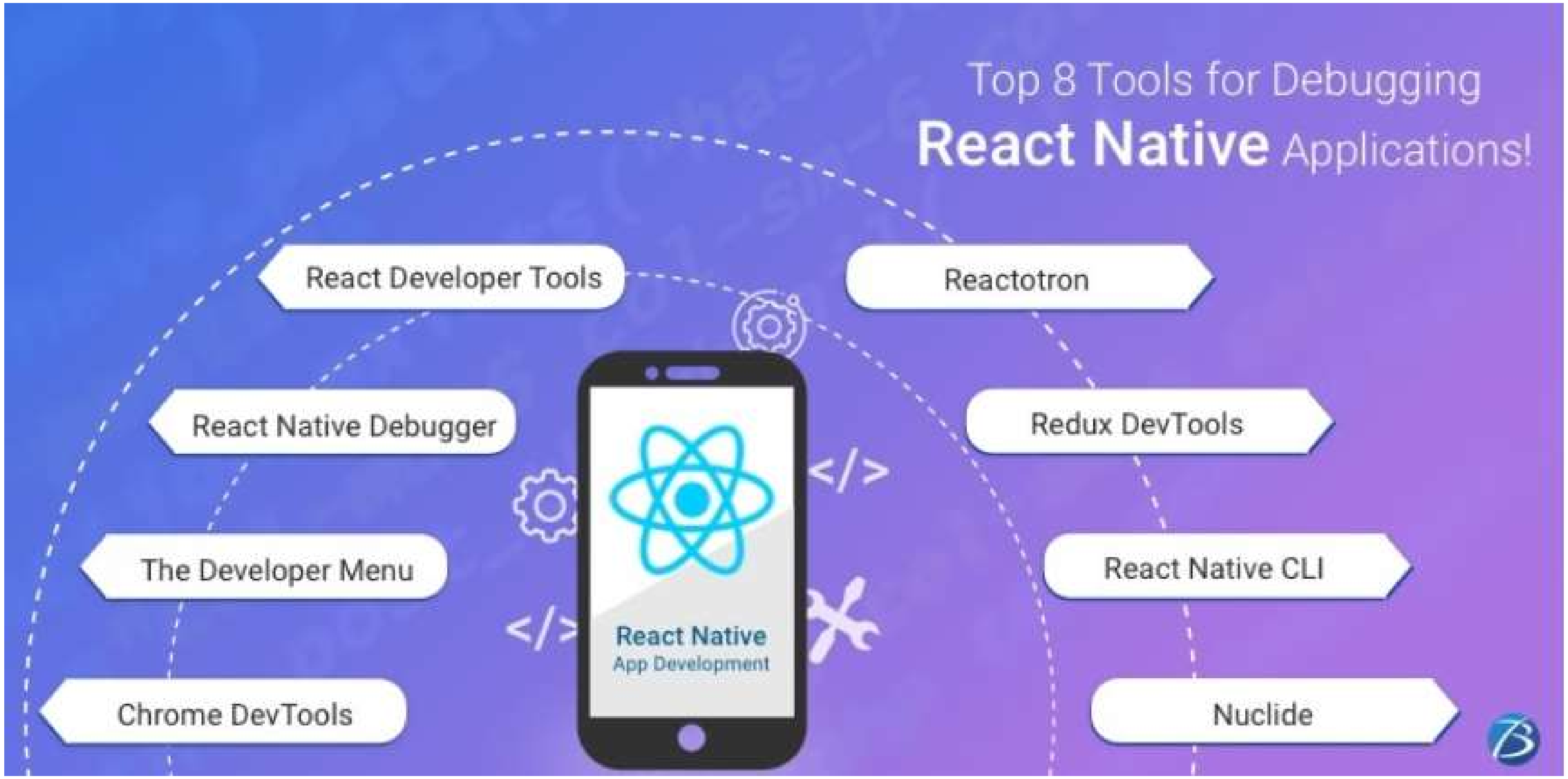


```
變數
  Local: addGoalHandler
    goalTitle: '2'
  > this: Object
  Closure (App)
  > _this: global {global: global, clearInterval: <accesso…
  > courseGoals: (0) []
  > removeGoalHandler: ƒ removeGoalHandler(goalId) {\n   …
  > setCourseGoals: ƒ ()
  > setIsAddMode: ƒ ()
  > Closure
  > Global
監看
  goalTitle.length === 0: false
呼叫堆疊
  Attach: Remote Process [0]                                PAUSED
    addGoalHandler                          App.js  18:5
    addGoalHandler        components/GoalInput.js  12:11
    _performTransitionSideEffects  node_modules/react-nativ…
    _receiveSignal  node_modules/react-native/Libraries/Pressa…
    onResponderRelease  node_modules/react-native/Libraries/Pr…
    invokeGuardedCallbackProd  node_modules/react-native/Lib…
    invokeGuardedCallback  node_modules/react-native/Librarie…
    invokeGuardedCallbackAndCatchFirstError  node_module…
    executeDispatch  node_modules/react-native/Libraries/Rend
已載入的指令碼
    C:/Users/jack/AppData/Local/Programs/Microsoft VS Code…
    child_process
  > components
    dgram
    index.js
中斷點
  ☐ Caught Exceptions
  ☐ Uncaught Exceptions
  ● ☑ App.js   14:5
  ● ☑ App.js   17:5
  ● ☑ App.js   26:5
  ● ☑ App.js   27:5
```

```javascript
import React, {useState} from 'react';  7.2K (gzipped: 3K)
import {StyleSheet, View, Button, FlatList} from 'react-native';

import GoalItem from './components/GoalItem';
import GoalInput from './components/GoalInput';

export default function App() {
  const [courseGoals, setCourseGoals] = useState([]);
  const [isAddMode, setIsAddMode] = useState(false);
  console.log('RE-RENDERING COMPONENT');
  console.log(courseGoals);

  const addGoalHandler = goalTitle => {
    if (goalTitle.length === 0) {
      return;
    }
    console.log('addGoalHanler', goalTitle);
    setCourseGoals(currentGoals => [
      ...currentGoals,
      {id: Math.random().toString(), value: goalTitle},
    ]);
    setIsAddMode(false);
  };

  const removeGoalHandler = goalId => {
```

```
[Info] 01-05 19:49:50.099  4099  4218 I ReactNativeJS: [ { id: '0.9789027017422114', value: '2' } ]

[Info] 01-05 19:51:52.186  4730  4822 I ReactNativeJS: Running "dd1" with {"rootTag":1}
01-05 19:51:52.245  4730  4822 I ReactNativeJS: RE-RENDERING COMPONENT
01-05 19:51:52.260  4730  4822 I ReactNativeJS: []

[Info] 01-05 19:52:35.995  4730  4822 I ReactNativeJS: RE-RENDERING COMPONENT
01-05 19:52:35.996  4730  4822 I ReactNativeJS: []

[Info] 01-05 19:52:38.903  4730  4822 I ReactNativeJS: 'addGoalHanler', '2'
01-05 19:52:38.907  4730  4822 I ReactNativeJS: RE-RENDERING COMPONENT
01-05 19:52:38.922  4730  4822 I ReactNativeJS: [ { id: '0.6656566368104079', value: '2' } ]

[Info] --------- beginning of system
```

Top 8 Tools for Debugging React Native Applications



Wrap Up

- So now we had a detailed look at the React Native debugging experience, at the React Native debugger and the different tools you have for finding and fixing errors, for analyzing your user interface, your component tree
- React Native debugger really is a cool tool for looking into your app, for setting breakpoints, viewing the console, viewing your component tree, viewing the styles you‘re using there and so much more. It really allows you to dive deeply into your application code, into your UI, into your logic and find out if everything is working the way it should work and you can even go in here and change certain things like the styling as you saw, to experiment with different settings and find out where you need to tweak your app for it to work correctly.
- Just simply debugging React Native in VS Code

MobileDrive

Maybe the Best Debugging Tool For You Is Yet to Come

Useful Resources & Links

- Debugging
  https://reactnative.dev/docs/debugging

- Chrome Dev Tools
  https://developers.google.com/web/tools/chrome-devtools/

- React Native Debugger
  https://github.com/jhen0409/react-native-debugger