

Debugging Instructions

- a. Printing to the console
 - i. Print Hello World (Should see the output on the terminal and the browser)
- b. Debugging using VS Code
- c. Debugging using Chrome

Debugging in Expo:

Few handy things to know before we start some coding

1. Printing to the console:

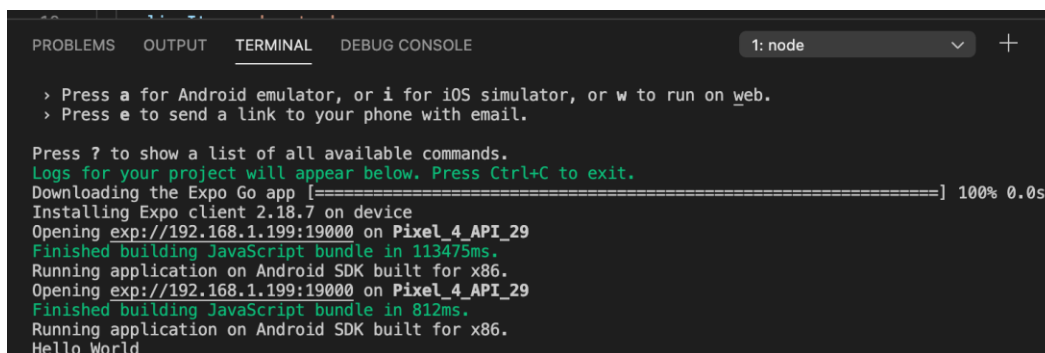
Printing to the console comes in handy whilst testing the codes. To print something onto the console, use this code.

```
console.log("Message to be displayed on the console");
```

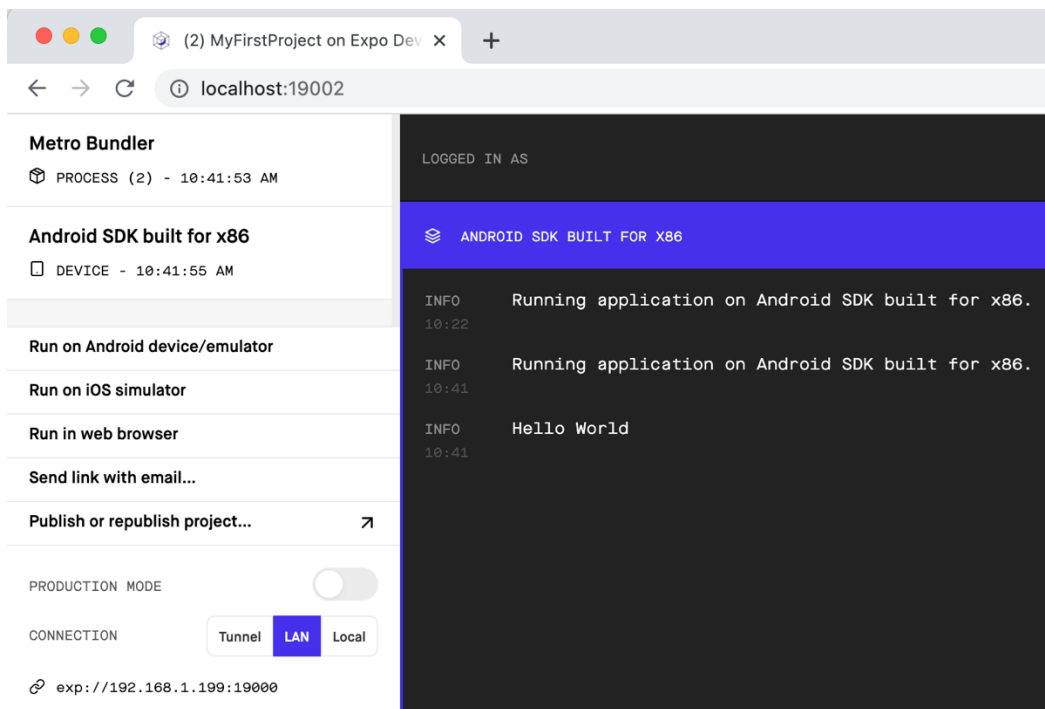
You could add a code to the App function and that should print the message on the console. Let's try "Hello World"

```
export default function App() {  
  console.log("Hello World");  
  return (  
    <View style={styles.container}>  
      <Text>Open up App.js to start working on your app!</Text>  
      <StatusBar style="auto" />  
    </View>  
  );  
}
```

You will see Hello World in the terminal in VS Code and also in the Browser Window

A screenshot of the VS Code terminal window. The terminal shows the output of running an Expo Go app. It starts with instructions to press 'a' for Android emulator, 'i' for iOS simulator, or 'w' to run on web, and 'e' to send a link to a phone. Then it shows the app running on a Pixel 4 API 29 device, with logs for downloading the Expo Go app, installing the Expo client, and building the JavaScript bundle. The final output is "Hello World".

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: node  
  
> Press a for Android emulator, or i for iOS simulator, or w to run on web.  
> Press e to send a link to your phone with email.  
  
Press ? to show a list of all available commands.  
Logs for your project will appear below. Press Ctrl+C to exit.  
Downloading the Expo Go app [=====] 100% 0.0s  
Installing Expo client 2.18.7 on device  
Opening exp://192.168.1.199:19000 on Pixel_4_API_29  
Finished building JavaScript bundle in 113475ms.  
Running application on Android SDK built for x86.  
Opening exp://192.168.1.199:19000 on Pixel_4_API_29  
Finished building JavaScript bundle in 812ms.  
Running application on Android SDK built for x86.  
Hello World
```

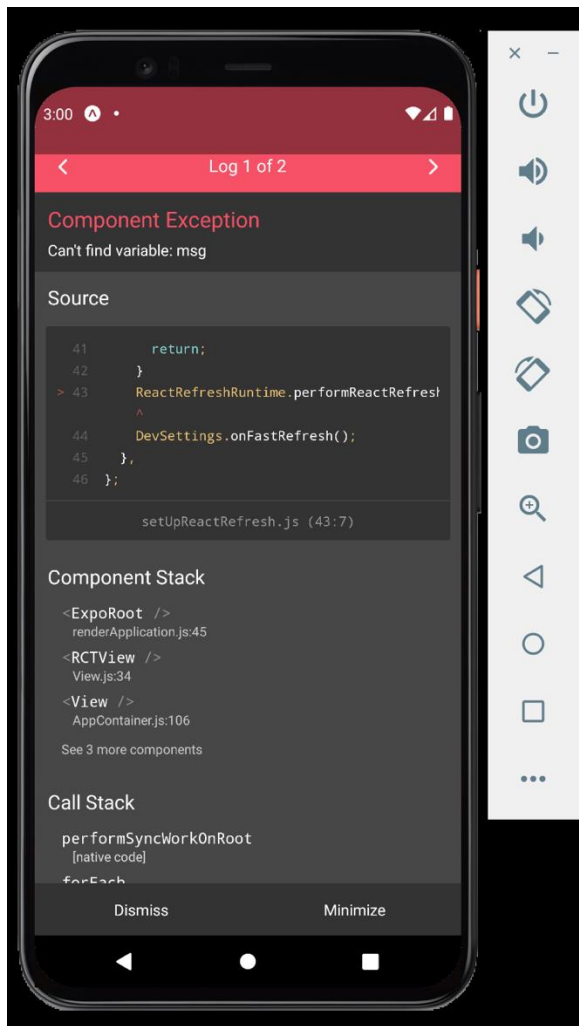


Let us create an error and explore the debugging options available to us

Onto the console, print a variable that has not been declared yet. Instead of Hello world, use the word 'msg'

```
console.log(msg);
```

When you save App.js, you should see a hot reloading and the app would crash with an error message

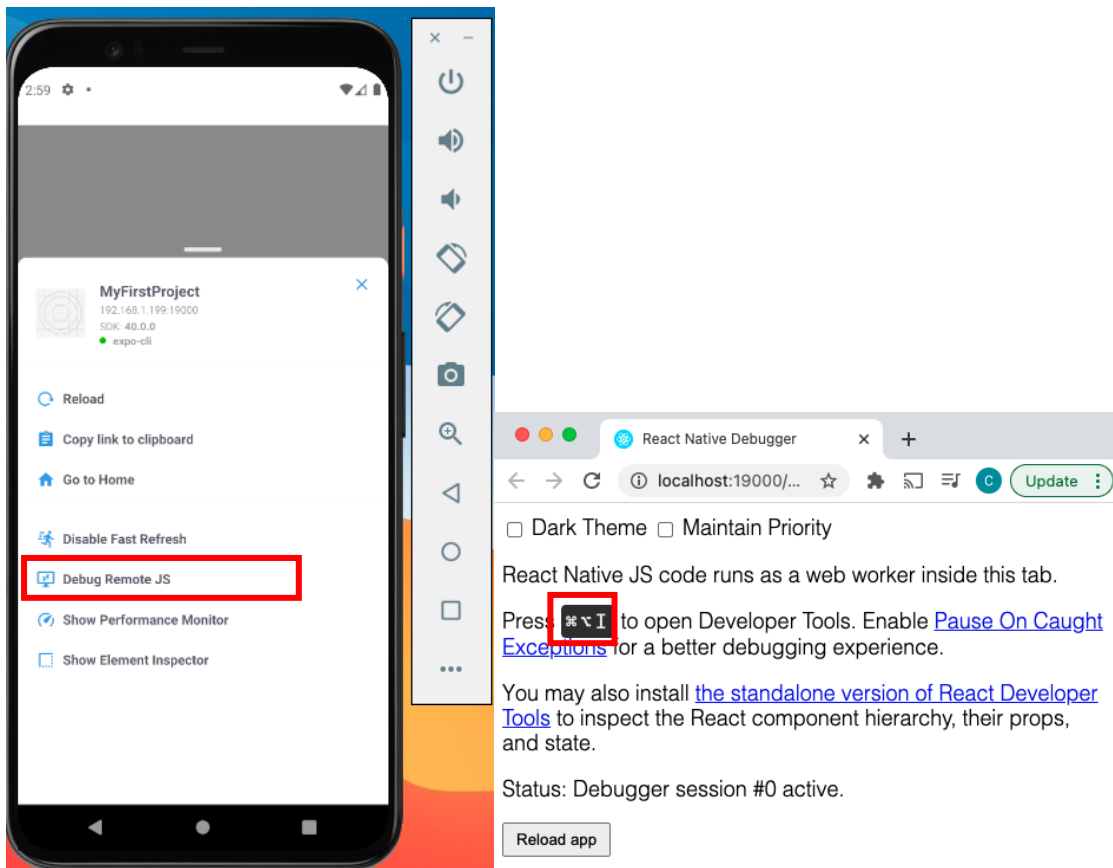


Knowing how to open the Developer Menu on the app is extremely important

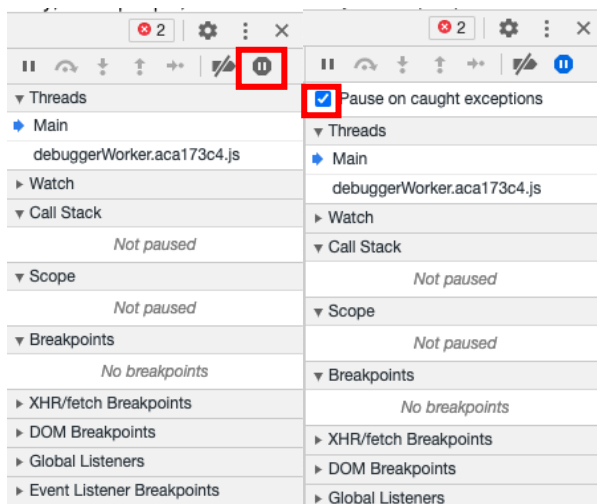
1. Shaking the device (if you are using one)
2. (Cmd+M)/(Cntrl+M) for Android Emulator
3. (Cmd+D)/(Cntrl+D) for iOS Simulator

2. Debugging via Chrome:

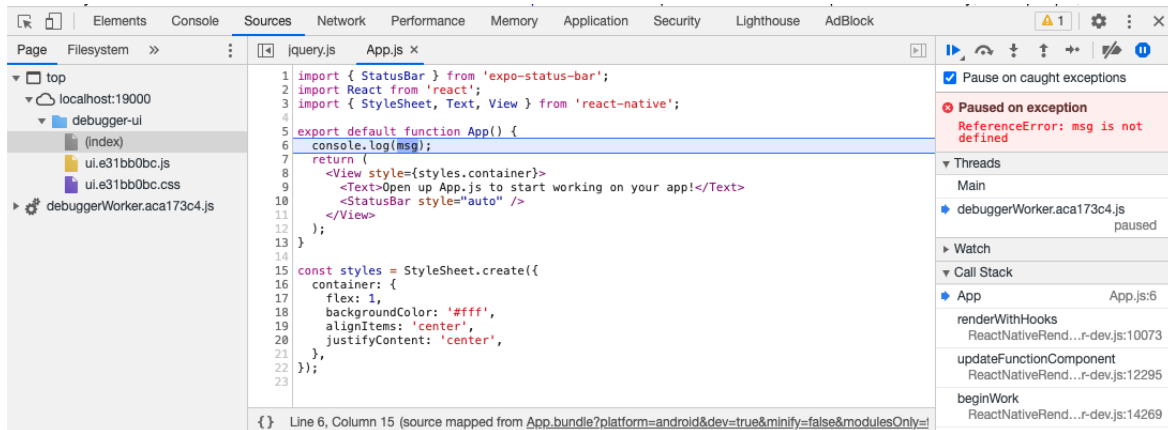
To debug the error on Chrome, the first step is to dismiss the message and open the developer menu. Sometimes this might not work since the app might not accept any inputs when it is in a crashed state (esp. in Android). Opening this option ahead of time, seems to work, if your Cmd/Cntrl+M option fails. To do that, comment the console.log line, save and you should see the app work. Now, use (Cmd+M)/(Cntrl+M) to view the developer menu. Click on Debug Remote JS and this will open a new tab on Chrome and keep this ready. Please remember to stop Remote debugging as this slows the execution of the app



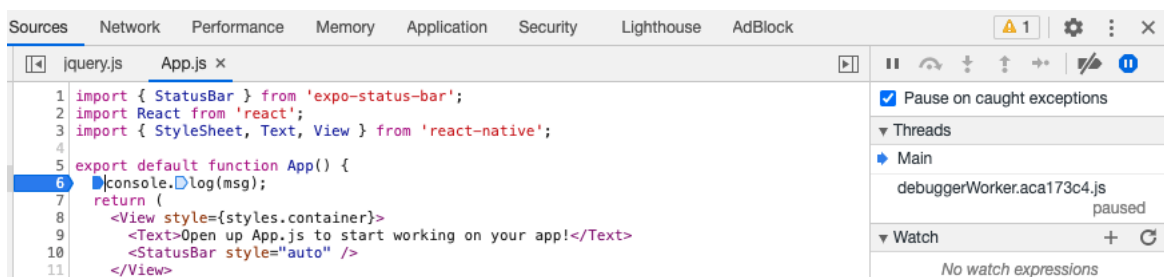
Open the Developer Tools (shortcut specified in Chrome) and that should open the Sources tab. On the right-corner you will see a Pause button. Select that and that would enable you to select a checkbox “Pause on caught Exceptions”



Now, uncomment the console.log line, save the app and you should see the exception has been caught by chrome and it pauses the execution with the error message



You can insert breakpoints and use Step over, Step in and Step out to debug your code.

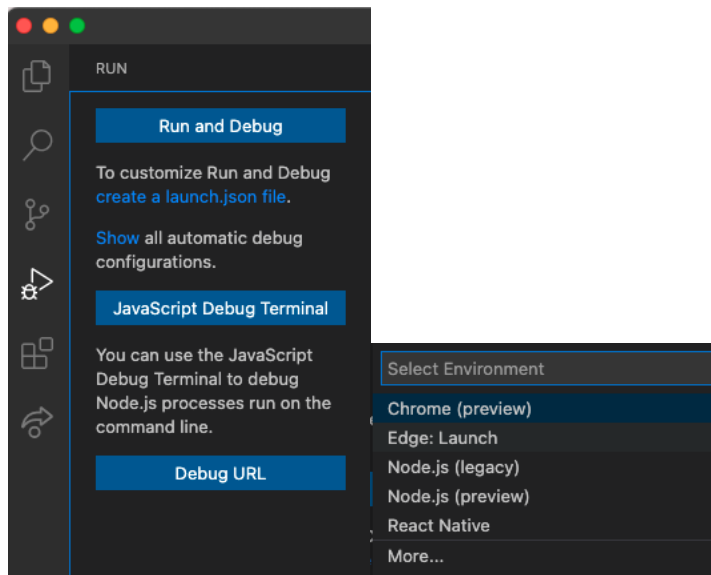


Please remember to stop Remote debugging

3. Debugging via VS Code:

Let us enable remote debugging again for the project.

To debug via VS Code, click on the Debugger option on the left and when you do it for the first time, it asks you to create a launch.json file to store the debug configurations for the project. When you click on Create a launch.json file, VS Code will prompt you for the environment. Please select React Native (if you have not installed the React Native Tools extension, this option may not be available)



By default, you will have an Android configuration, deselect that and select Attach to Packager configuration. You can always add configurations from launch.json file

```
Js App.js {} launch.json •
.vscode > {} launch.json > JSON Language Features > [ ] configurations
1 {
2   // Use IntelliSense to learn about possible attributes.
3   // Hover to view descriptions of existing attributes.
4   // For more information, visit: https://go.microsoft.com/fwlink/?linkid=8303
5   "version": "0.2.0",
6   "configurations": [
7
8     {
9       "name": "Attach to packager",
10      "cwd": "${workspaceFolder}",
11      "type": "reactnative",
12      "request": "attach"
13    }
14  ]
15 }
```

Add Configuration...

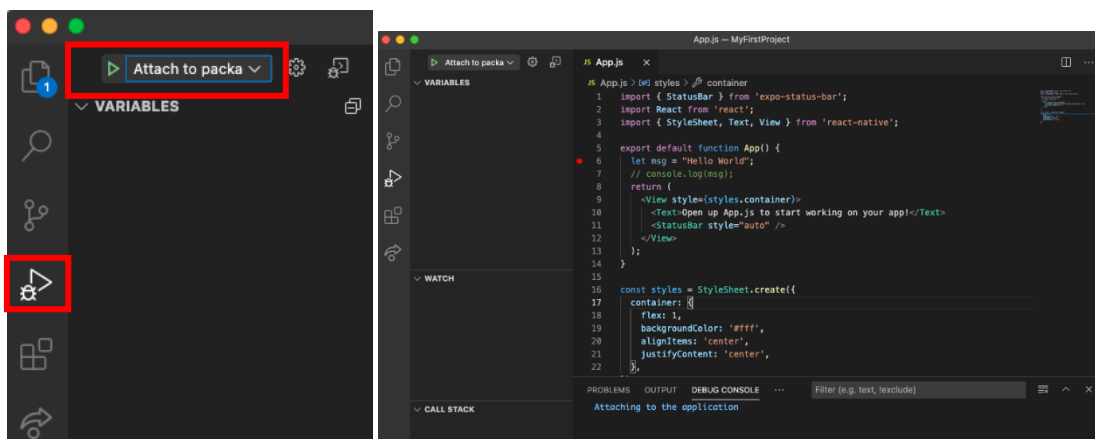
You can add breakpoints to the App.js file. Declare a variable called msg and set it to Hello World. Also, add a breakpoint on that line

```

JS App.js
JS App.js > App
1 import { StatusBar } from 'expo-status-bar';
2 import React from 'react';
3 import { StyleSheet, Text, View } from 'react-native';
4
5 export default function App() {
6   let msg = "Hello World";
7   // console.log(msg);
8   return (
9     <View style={styles.container}>
10      <Text>Open up App.js to start working on your app!</Text>
11      <StatusBar style="auto" />
12    </View>
13  );
14

```

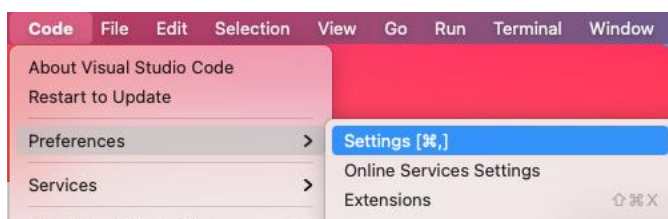
To run this app in a debug mode, go to Debug panel-> you should see your configuration at the top (Attach to packager) and click on the play button. You can view the status in the debug console



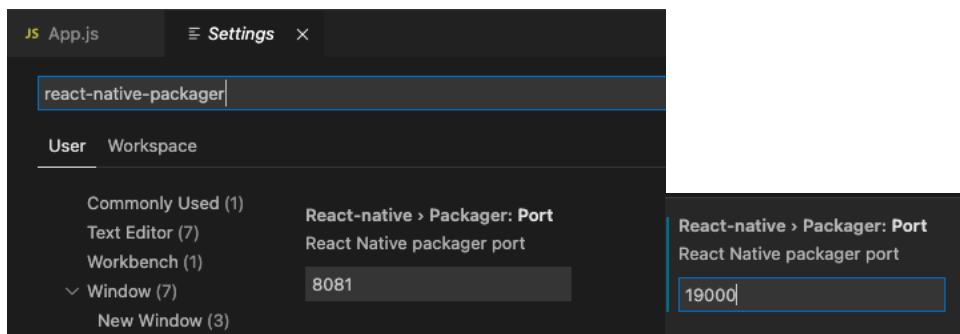
This could be because of the port number. When you open the remote JS debugger, the tab that opens on chrome will have the port number.

For instance, on my machine it is <http://localhost:19000/debugger-ui/>

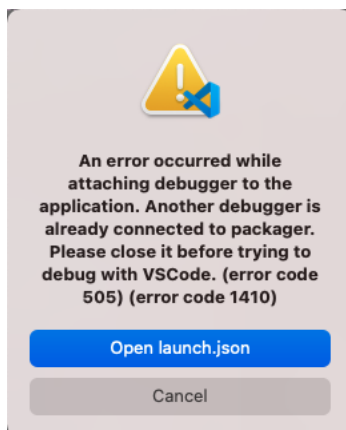
19000 is the port number. This has to be added to Code-> Preferences-> Settings



Search for react-native-packager and change the port number to the number on your machine (in my case, 19000)



If you try again, it might come up with another error message that another debugger is already connected (This means that the Chrome Window has to be closed)



Once the window is closed, use the debug panel and click on play again and you should see in the debug console that the connection has been established. To debug, reload the app and you should see the app stop at the breakpoint. Now, step over the line and see the variable's value change. You can also add variables to Watch to see how the values change. You can disconnect and stop the debugger

