

Slot 5 – Debug

Các cách dùng:

C1 – console.log trong code

C2 - Xem phần console trong chrome

C3 – Dùng breakpoint: xác điểm dừng của chương trình → tìm lỗi

Yêu cầu:

- update nodejs

brew update

brew upgrade node

- cài debugger

brew install -cask react-native-debugger

=====

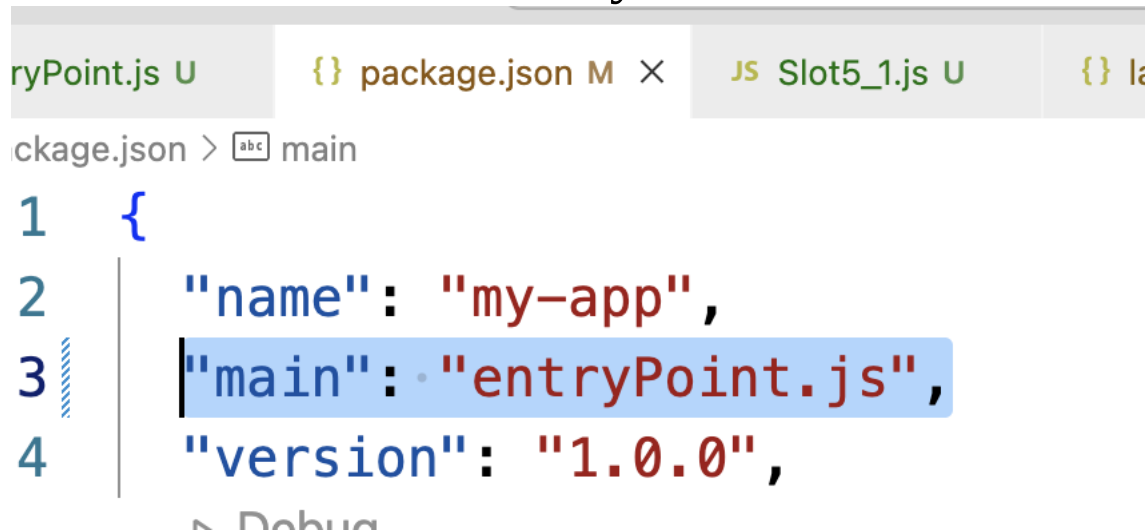
Cấu hình:

B1- tạo EntryPoint.js (cùng cấp với Package.json)



```
JS entryPoint.js U × {} package.json M JS Slot5_1.js U {} launch.json 1, U JS Ap
JS entryPoint.js
1 import { registerRootComponent } from "expo";
2 import App from "./App";
3 registerRootComponent(App);
```

B2 – cấu hình main cho EntryPoint



The screenshot shows the VS Code editor with the `package.json` file open. The file content is as follows:

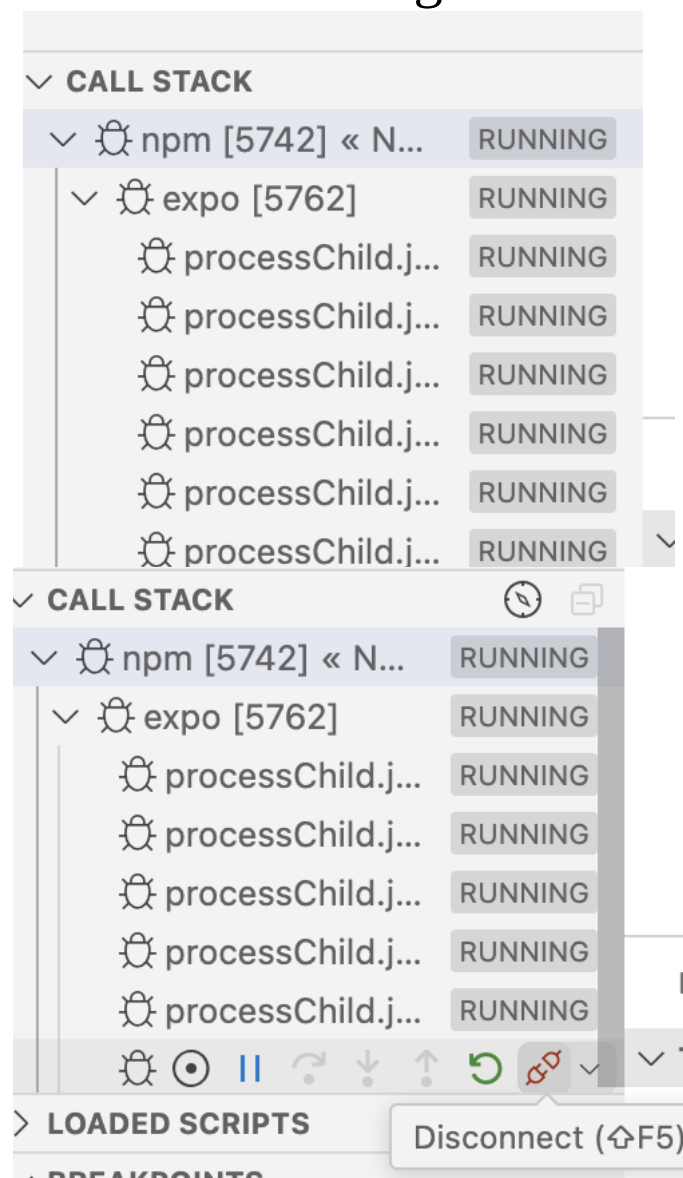
```
1 {  
2   "name": "my-app",  
3   "main": "entryPoint.js",  
4   "version": "1.0.0",  
}
```

The `"main": "entryPoint.js",` line is highlighted in blue. The editor tabs at the top show `ryPoint.js U`, `package.json M`, `JS Slot5_1.js U`, and `package.json`.

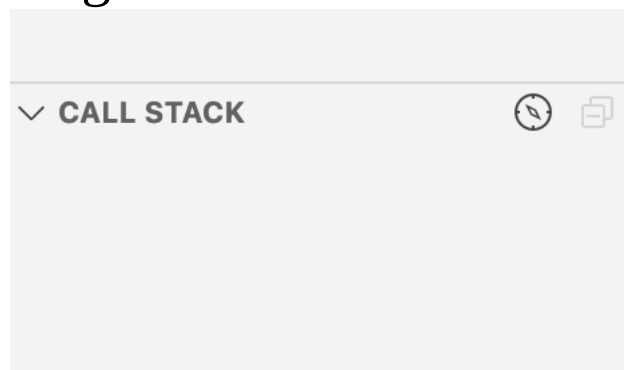
Các bước chạy:

B1 – npm start, nhấn W

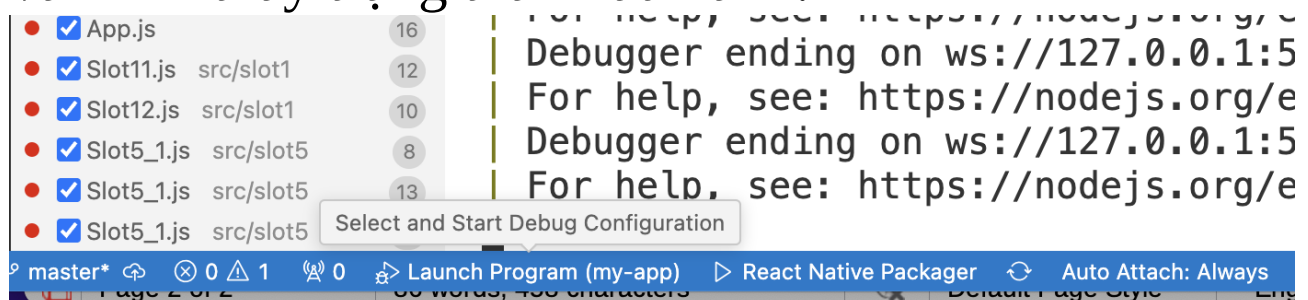
B2 – Tắt tất các tiến trình trong call stack



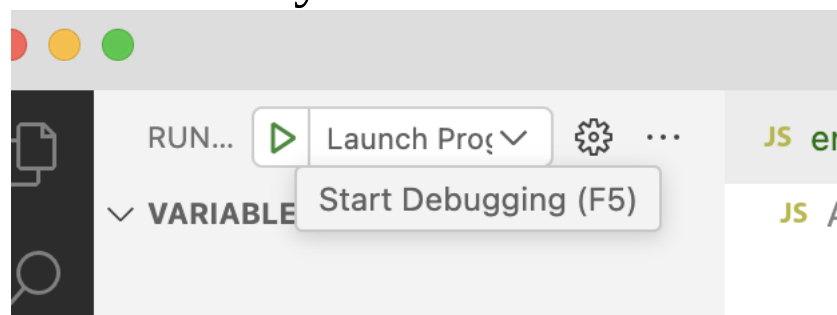
Kết quả cuối cùng:



và Nhìn thấy trạng thái màu xanh:



B3 – Nhấn button Play



Lúc này

1 trình duyệt mở ra, ta gõ localhost:8081 và enter
=> nếu chương trình chạy vào breakpoint thì tức là ta
cấu hình đúng

=====

`onPress={this.increment()}`

=> không cần `press` => nó tự động gọi sau mỗi lần render

`onPress={()=>this.increment()}`

=> phải `press` thì mới gọi hàm

Giải thích: khi `press` → gọi đến 1 hàm → hàm này gọi đến hàm `increment`

=> đây là cách gọi callback trong hàm mũi tên

=====

II. hàm mũi tên

- dùng để truyền callback vào các sự kiện: `onPress`, `onClick`

- dùng để truyền tham số cho các hàm sự kiện

- Không có `this` => không cần bind

```
1  import { Button } from "react-native";
2
3  //Truyen ham khong co tham so vao su kien
4  const handlePress = () =>{
5    console.log("Ban vua click button");
6  };
7  <Button title="Press me" onPress={handlePress}/>
8  //truyen ham co tham so vao su kien
9  const handlePress1 = (id)=>{
10   console.log(`Ban vua click lan thu ${id}`);
11 };
12 <Button title="Click me" onPress={()=>handlePress1(1)} />
```

- dùng để xử lý sự kiện trong các hook `useState`, `useEffect`

```

import { useState } from "react";
import { Button } from "react-native";

const MyCom = () =>{
  //code
  const [count, setCount]=useState(0);
  const increment={()=>{
    setCount(count+1);
  }};
  //layout
  return(
    <Button title="Press me" onPress={()=>increment()} />
  );
};

```

=====

3. Khi nào không sử dụng hàm mũi tên???

-Mỗi khi hàm mũi tên được gọi → render lại → mất nhiều bộ nhớ

=> cần tối ưu hóa trong việc sử dụng callback

hoặc cần tách hàm mũi tên ra bên ngoài

- không dùng từ khóa **new** với hàm mũi tên

=====

4. Truyền tham số cho hàm mũi tên

4.1 Tham số là 1 biến

```
1 ✓ //4.truyen tham so cho ham mui ten
2 //4.1 tham so la 1 bien
3 ✓ const inTen = (name)=>{
4   |   console.log(`Xin chao ${name}`);
5   | };
6   inTen("Nguyen Van An");
```

4.2 tham số là 1 đối tượng

```
//-----tham so la 1 doi tuong (full)
const inDoiTuong = (person)=>{
  |   console.log(`Name: ${person.name}, Age: ${person.age}`);
  | };
const person={name: "An", age: 20};
inDoiTuong(person);|
```

4.3. tham số là 1 mảng

```
//-----tham so la 1 mang (full)
const tinhTongMang = (numbers) =>{
  |   return numbers.reduce((total,num)=>total+num,0);
  | };
const numbers=[1,2,3,4,5];
console.log(tinhTongMang(numbers));//15
```

4.4 tham số là 1 đối tượng với cấu trúc phân rã:
ý nghĩa: khi chỉ cần dùng 1 vài thuộc tính của đối tượng có 1000 thuộc tính => hãy dùng đối tượng với cấu trúc phân rã

```
//-----su dung doi tuong voi cau truc phan ra
const inThuocTinh = ({name,age})=>{
  console.log(`Name: ${name}, Age: ${age}`);
};
const person1 = {name: 'NVA',age:22};
inThuocTinh(person1);
```

4.5. Tham số là mảng với cấu trúc phân rã:

ý nghĩa: khi chỉ cần sử dụng 1 vài phần tử của mảng có 1000 phần tử => sử dụng cấu trúc phân rã mảng

```
//----- su dung mang voi cau truc phan ra
const inPhanTuMang = ([first,second])=>{
  console.log(`First: ${first}, Second: ${second}`);
};
const mang=[10,20,30,40,50,60,70,80];
inPhanTuMang(mang);//10,20
```

4.6 Truyền callback cho hàm mũi tên

Hàm gọi đến hàm ở bên trong

```
//----- tham so la 1 ham callback
const executeCallback = (callback)=>{
  console.log("Truoc khi goi callback");
  callback();
  console.log("Sau khi goi callback");
};
executeCallback(()=>console.log("Day la ham callback"));
```

===

Hàm callback:

- hàm callback là hàm được truyền vào hàm khác như là 1 tham số (là tham số của 1 hàm khác)
- Khi bạn muốn hành động A kết thúc thì gọi đến hành động B => B chính là callback

- DÙNG khi xử lý đồng bộ, không đồng bộ, API
=====