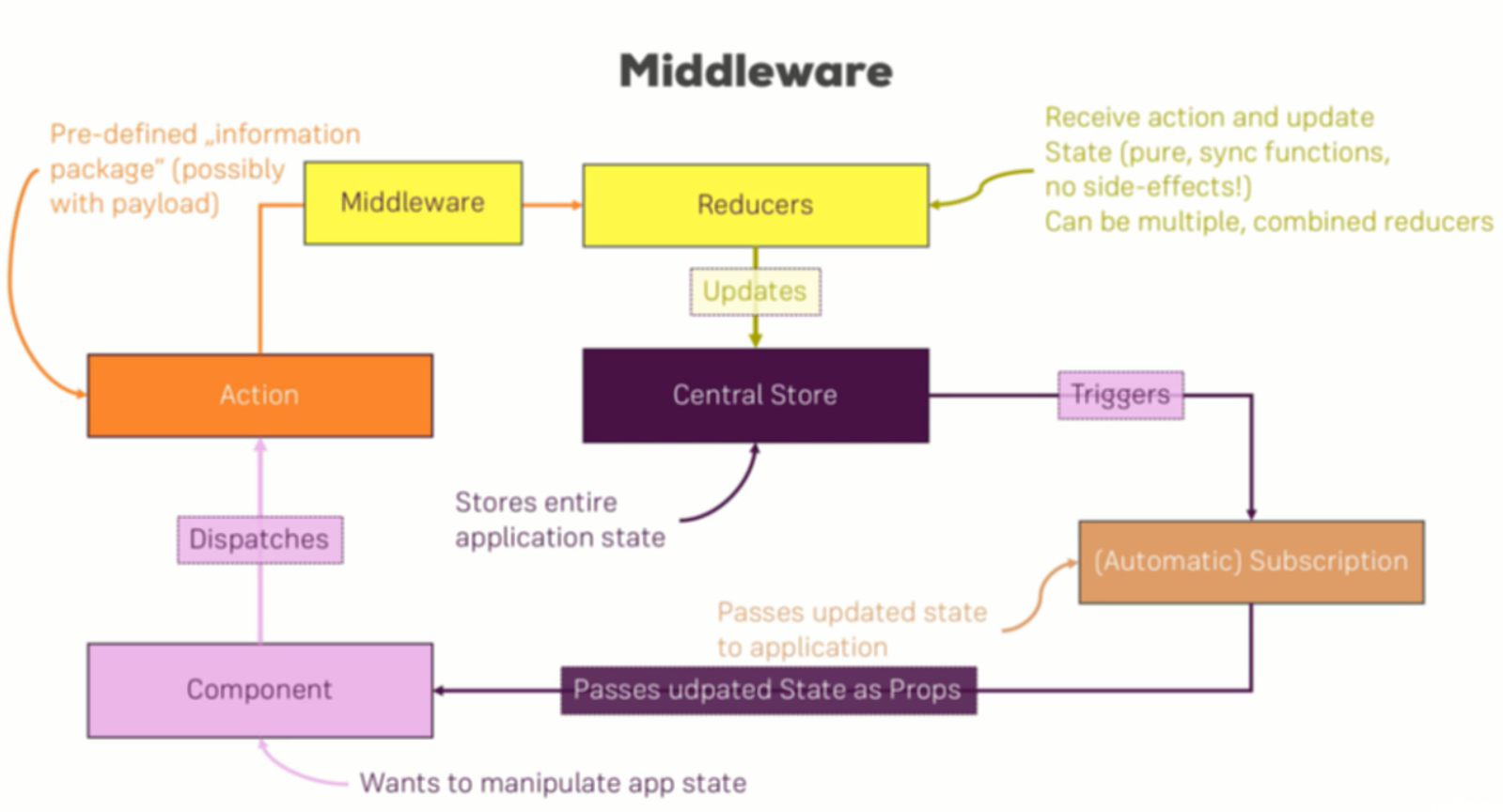
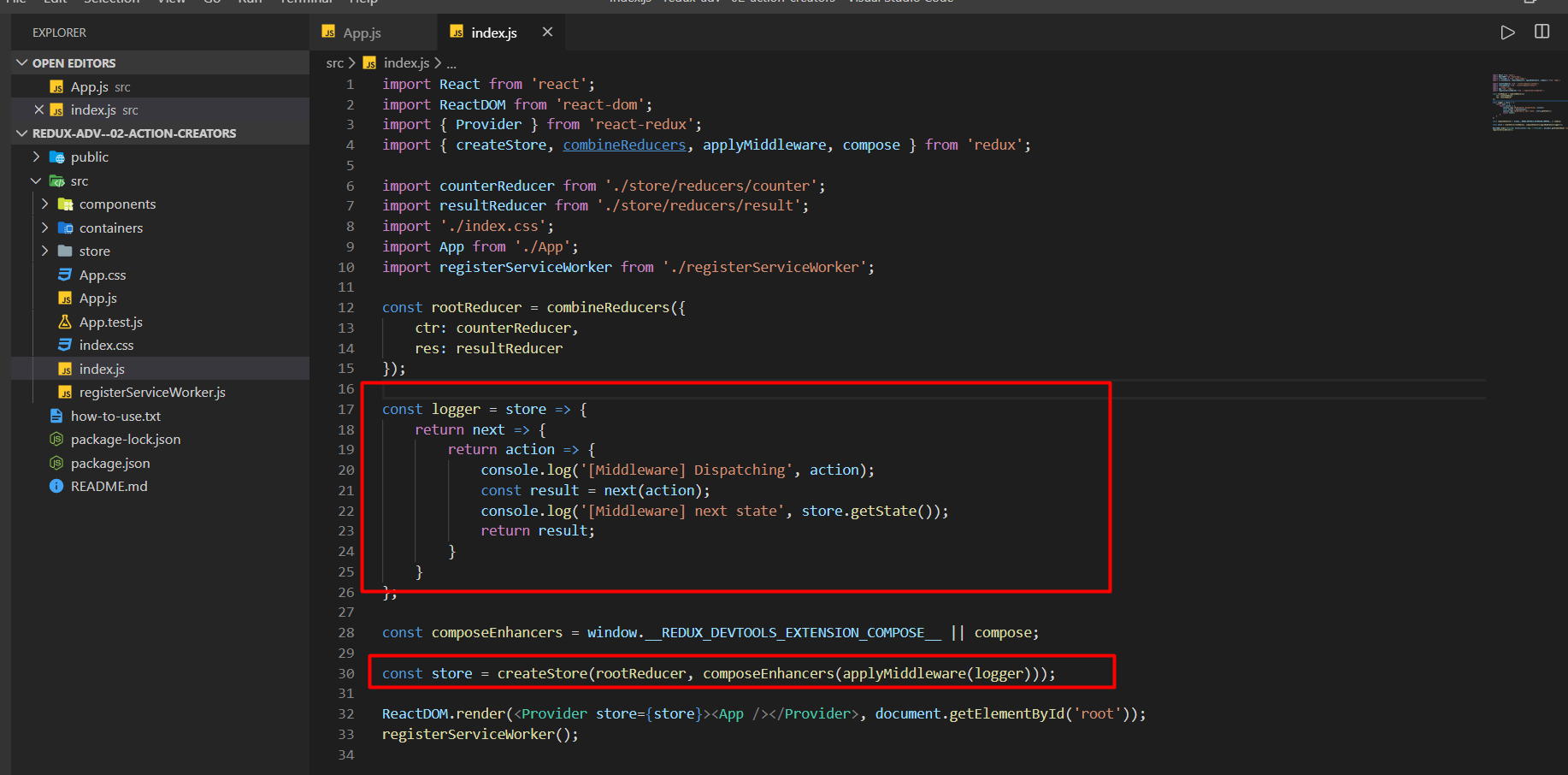
2. Adding Middleware



Middleware như cái filter giữa action chạy qua reducer vậy

Add thêm điều kiện vào để action vượt qua điều kiến đó sẽ đến update reducer





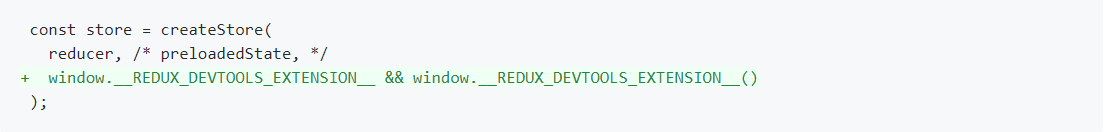
Viết hàm tương tự như trên, apply middleware đó vào store

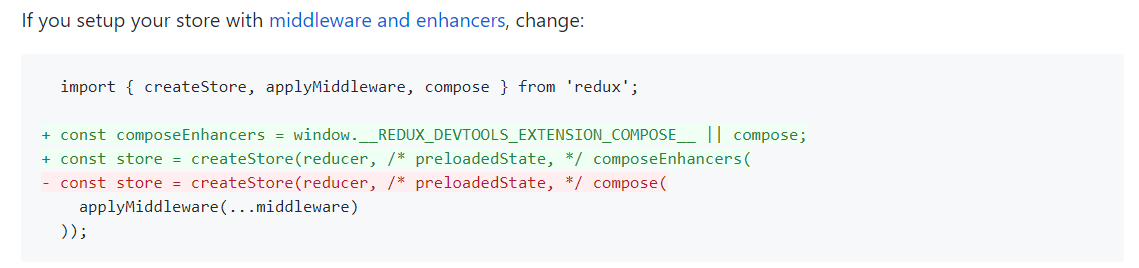
Có thể apply nhiều apply vào store, cách nhau bằng dấu ,

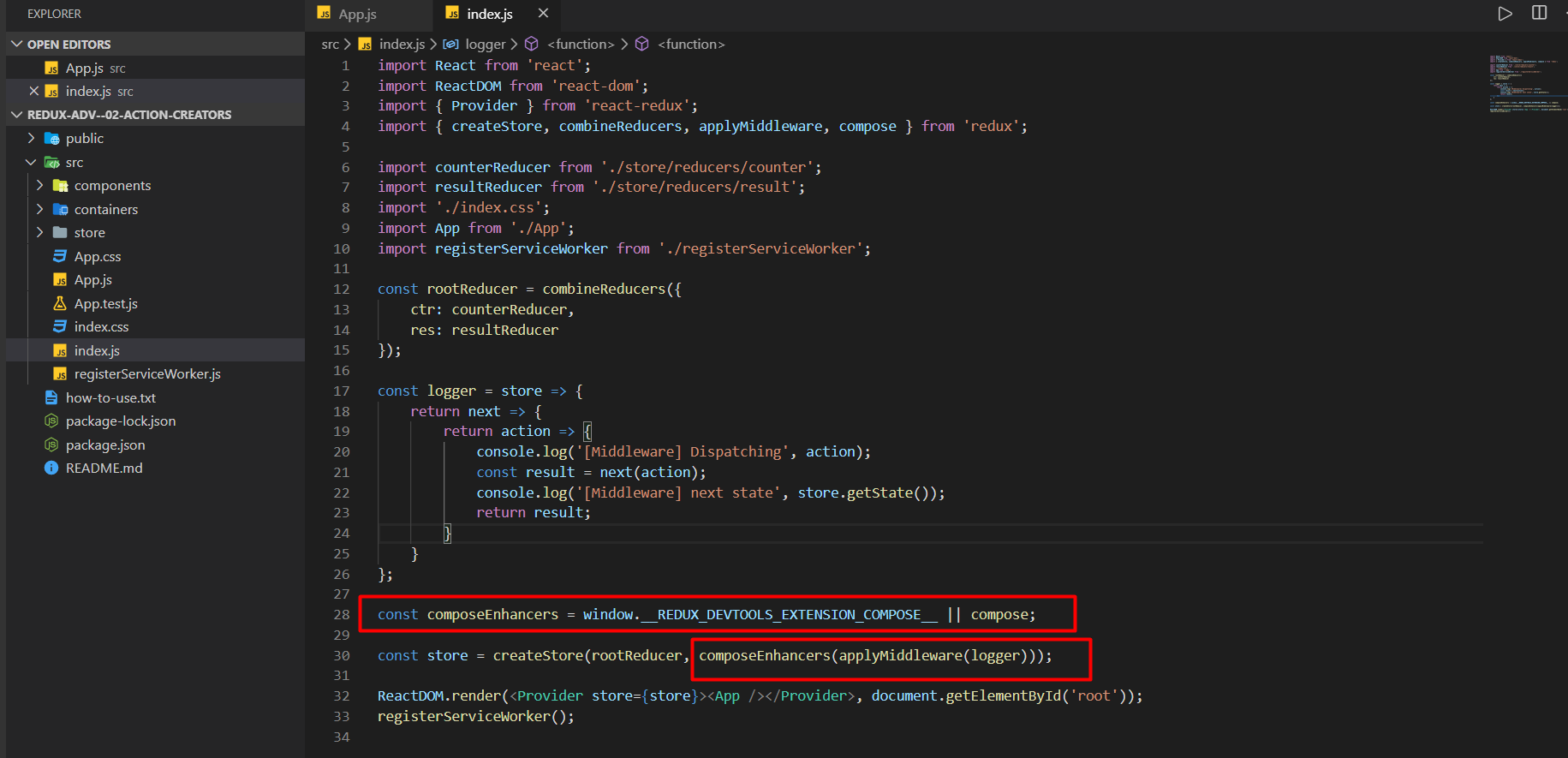
Như đã thấy thì middleware ở trên đứng giữa action và reducer -> có thể đưa điều kiện vào đây

3. Using the Redux Devtools

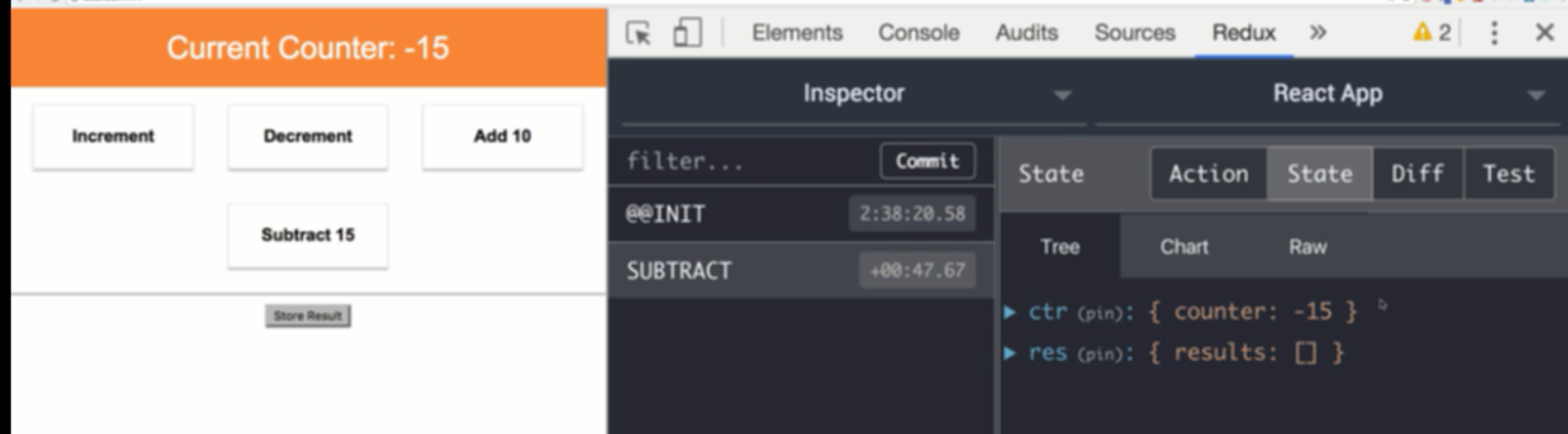
<https://github.com/zalmoxisus/redux-devtools-extension>

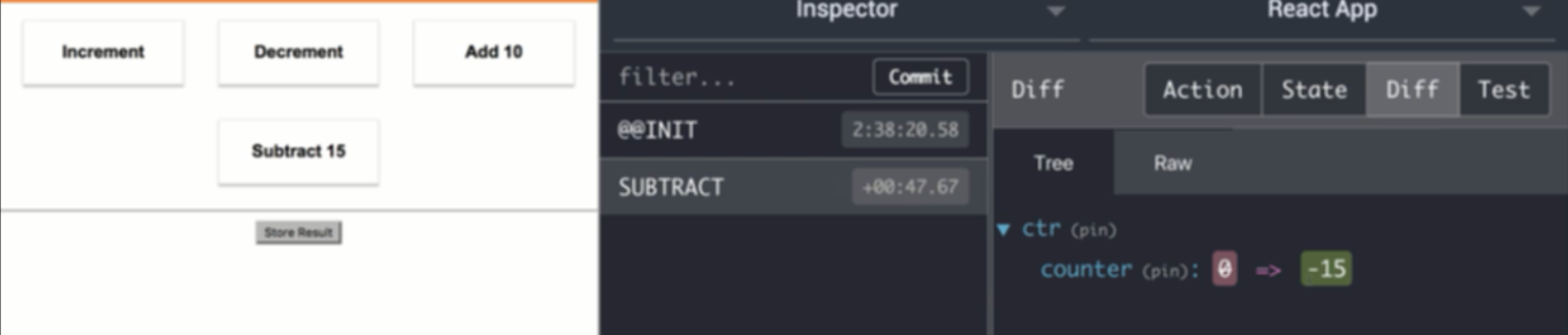




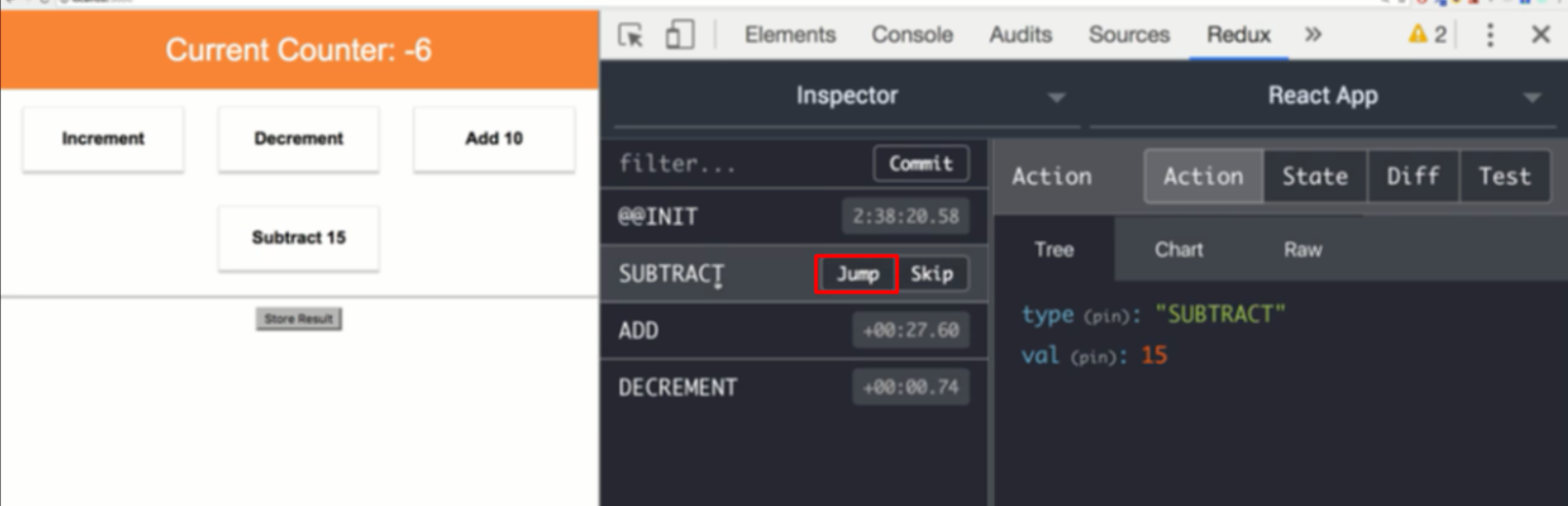


Setup để có thể dùng được redux devtool

Khi thực hiện hành động liên quan đễn store thì action sẽ hiện lên cho phép xem các thông tin về redux



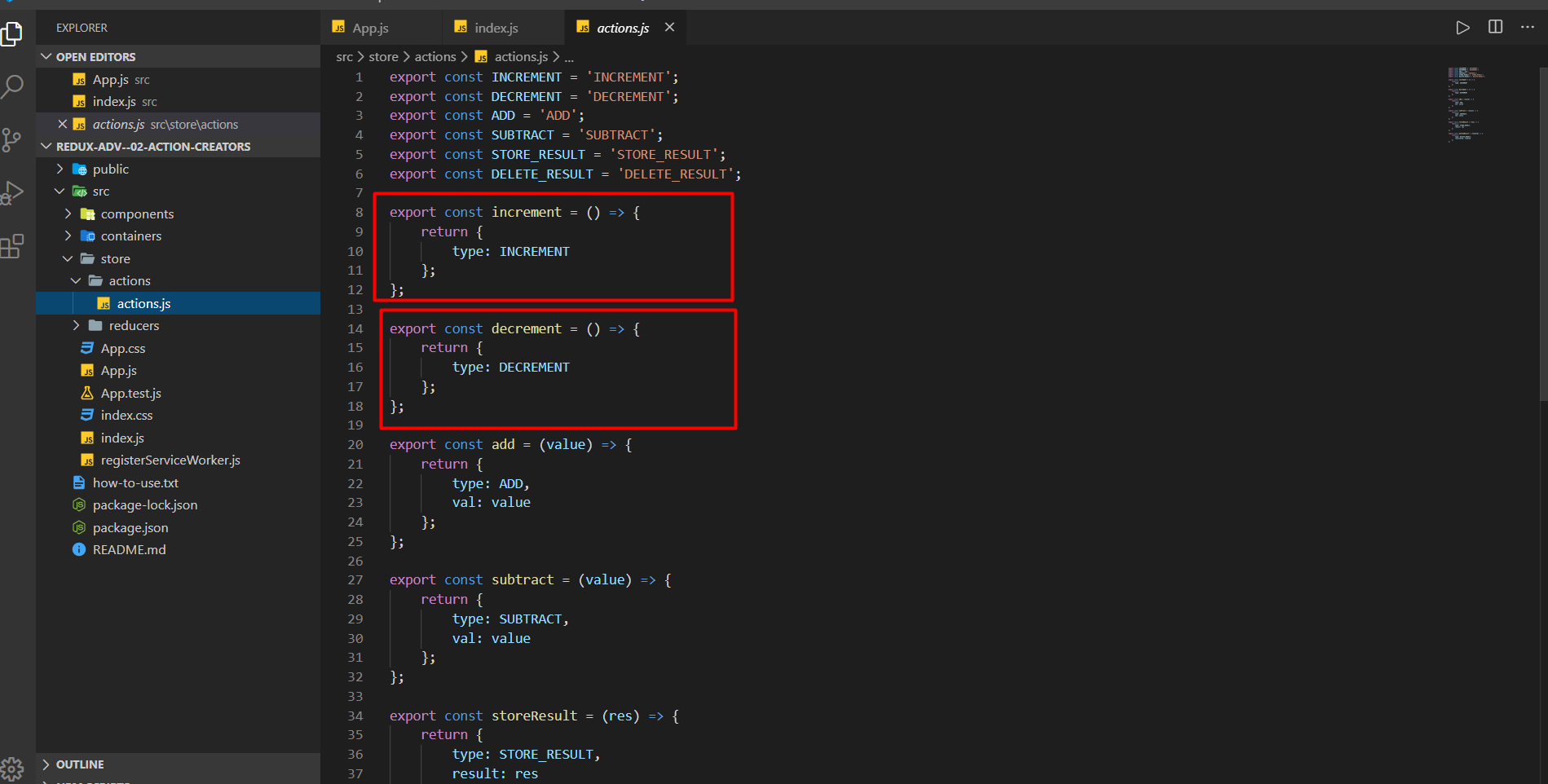
Xem state chuyển đổi ntn nữa



Trở về từng thời điểm được luôn

Skip là bỏ các action đó ra như khi chưa bấm action đó

4. Executing Asynchronous Code – Introduction

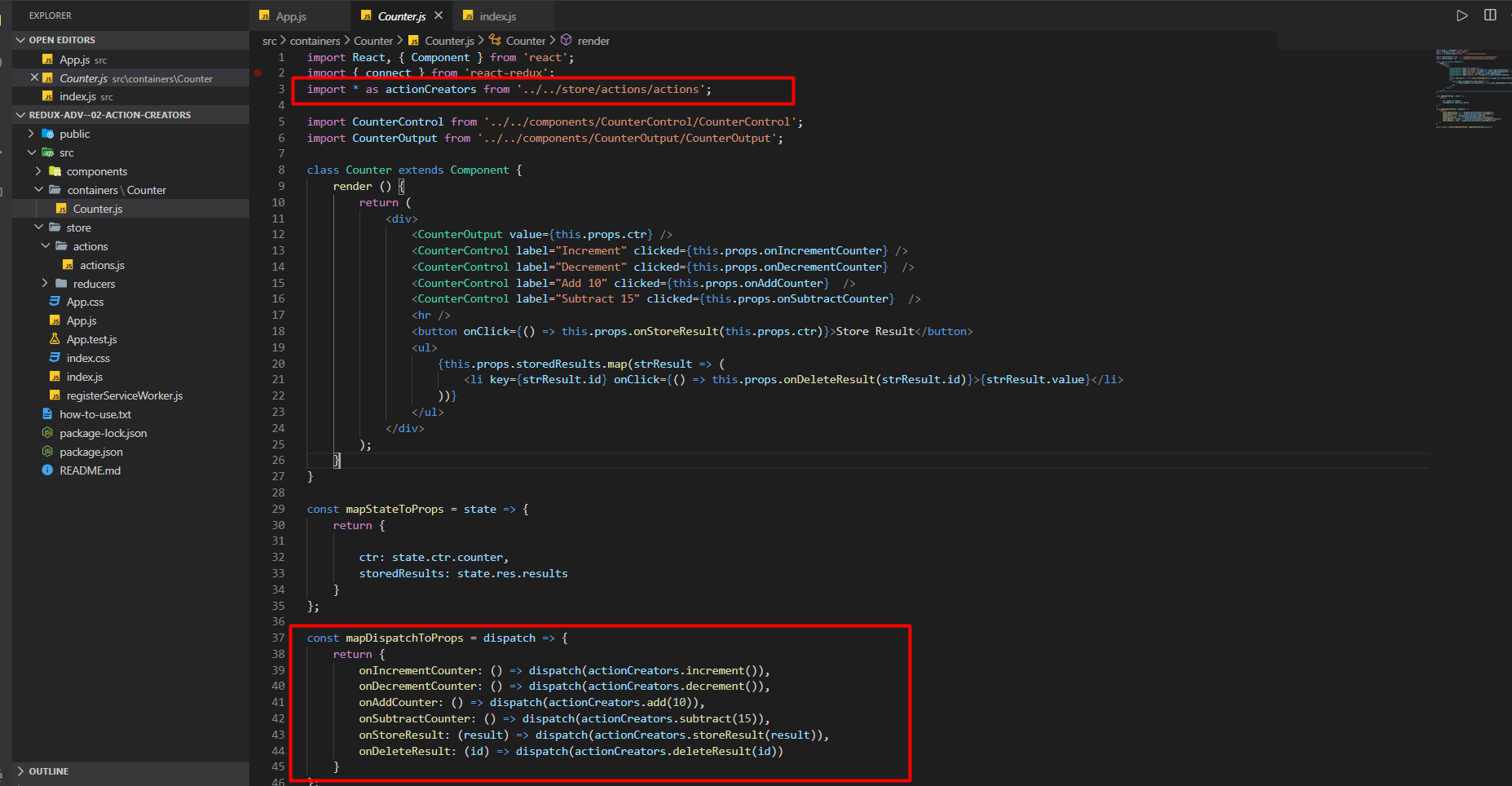


Tạo các action là các hàm

Và có thể đưa giá trị vào các hàm này, để khi gọi hàm trên component thì truyền giá trị trong component đó vào

Ví dự là truyền biến từ reduce này sang reduce khác

Hoặc ví dụ: cộng thêm 1 giá trị vào tổng, giá trị đó được nhập từ ô input trên component -> truyền giá trị đó vào action để đưa qua reduce add vào state



Import vào và sử dụng các action trong dispatch, những có có truyền biến thì cần đưa biến vào

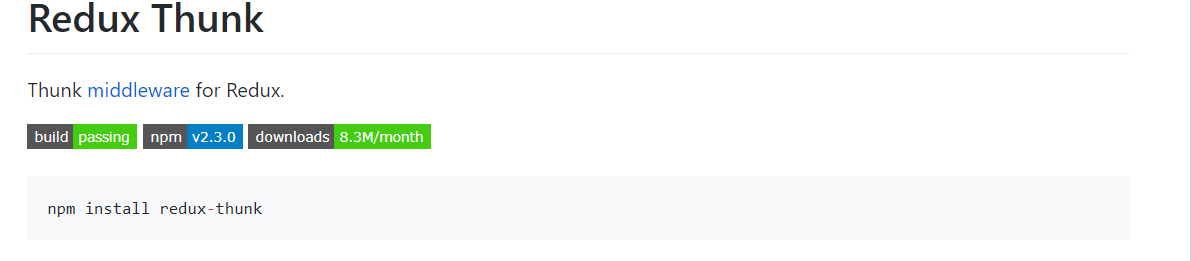
7. Handling Asynchronous Code

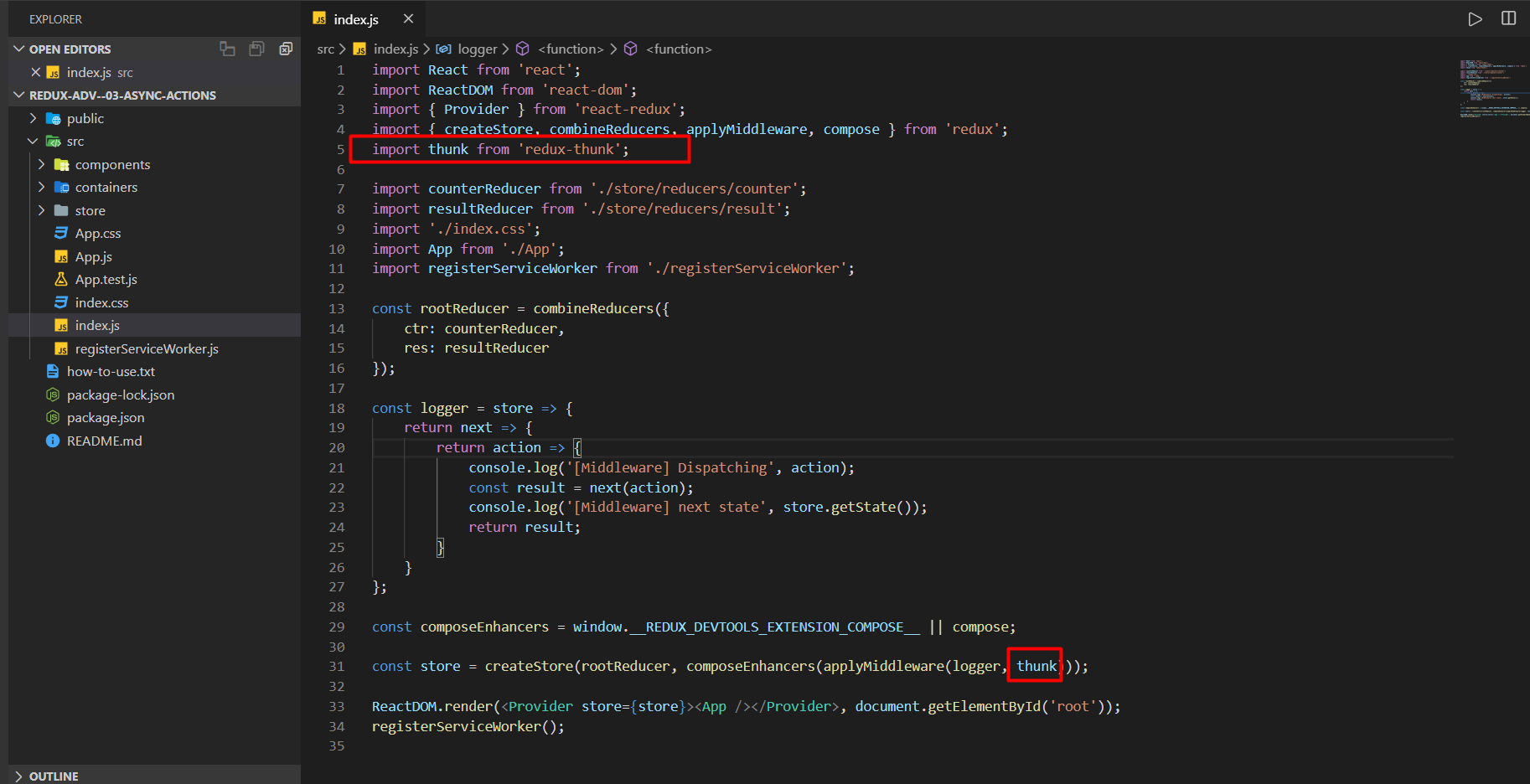
<https://github.com/reduxjs/redux-thunk>

dùng middleware redux-thunk

action khi được create sẽ không return cái action đó mà sẽ return 1 cái funtion dispatch action

* Khi đó mới có thể chạy async trên 1 cái action





Import thunk vào để sử dụng

Phần này lấy từ khóa zero to mastery

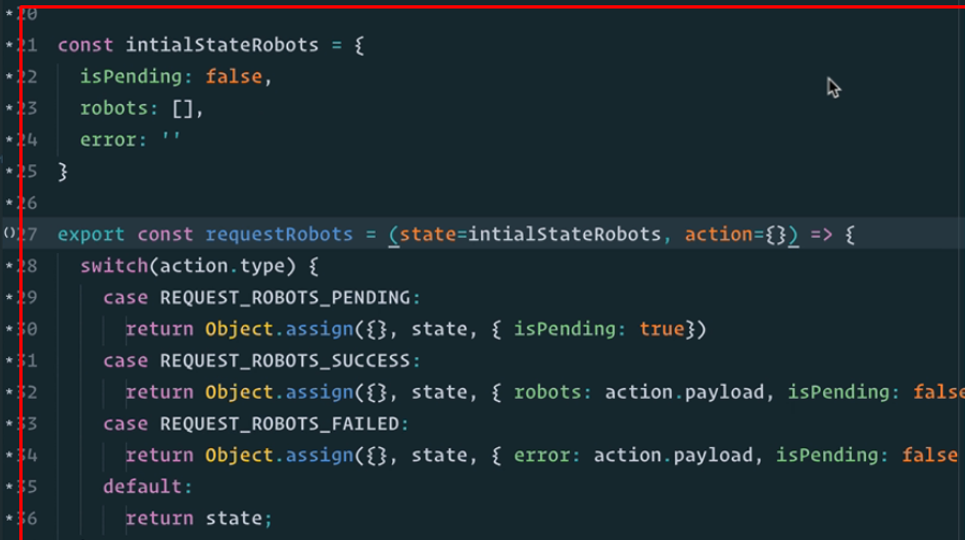


Cấu trúc hàm như trên, mỗi lệnh dispatch sẽ thự hiện action đó

Khi bắt đầu gửi -> đưa biến pending

Gửi xong: đưa biến success và đưa data ra

Gử lỗi: đưa error ra

reducer

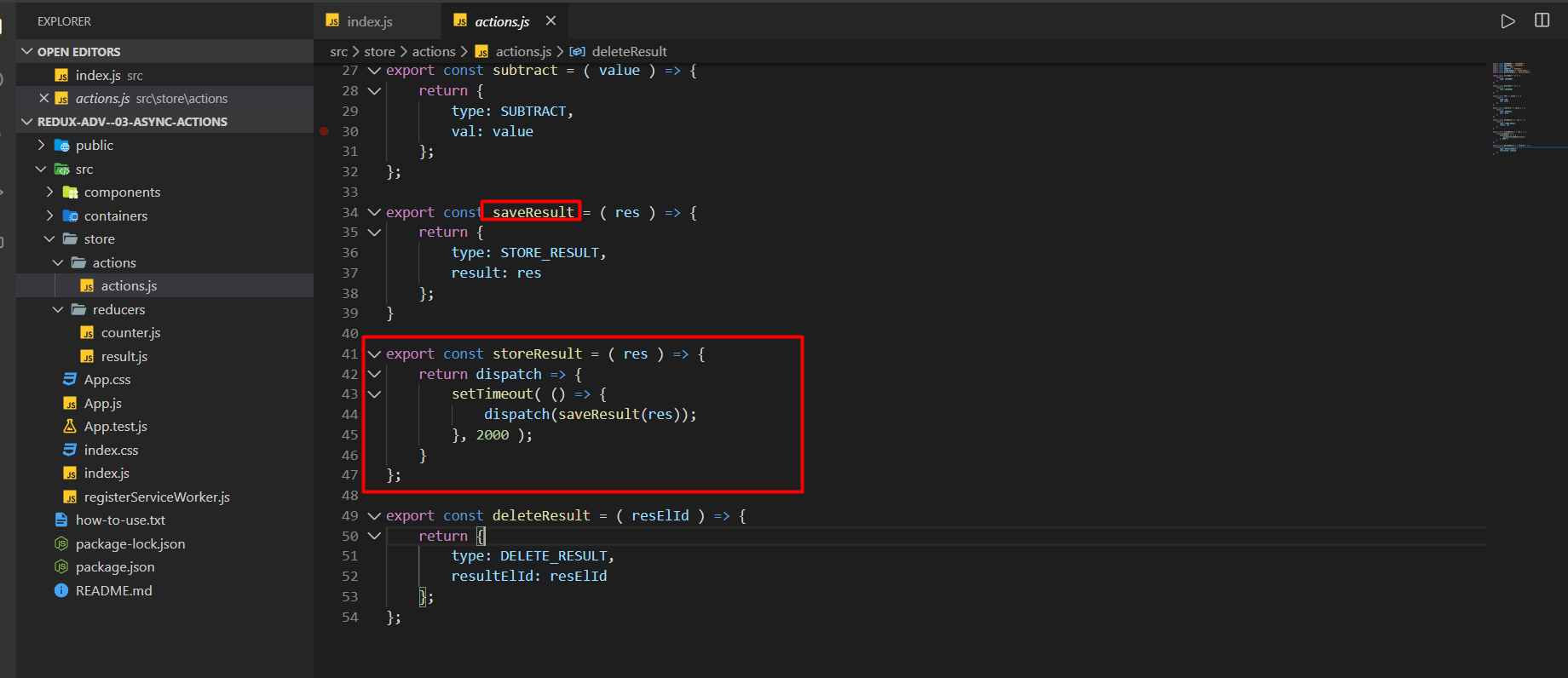


Xài

Hoặc xài theo cách này



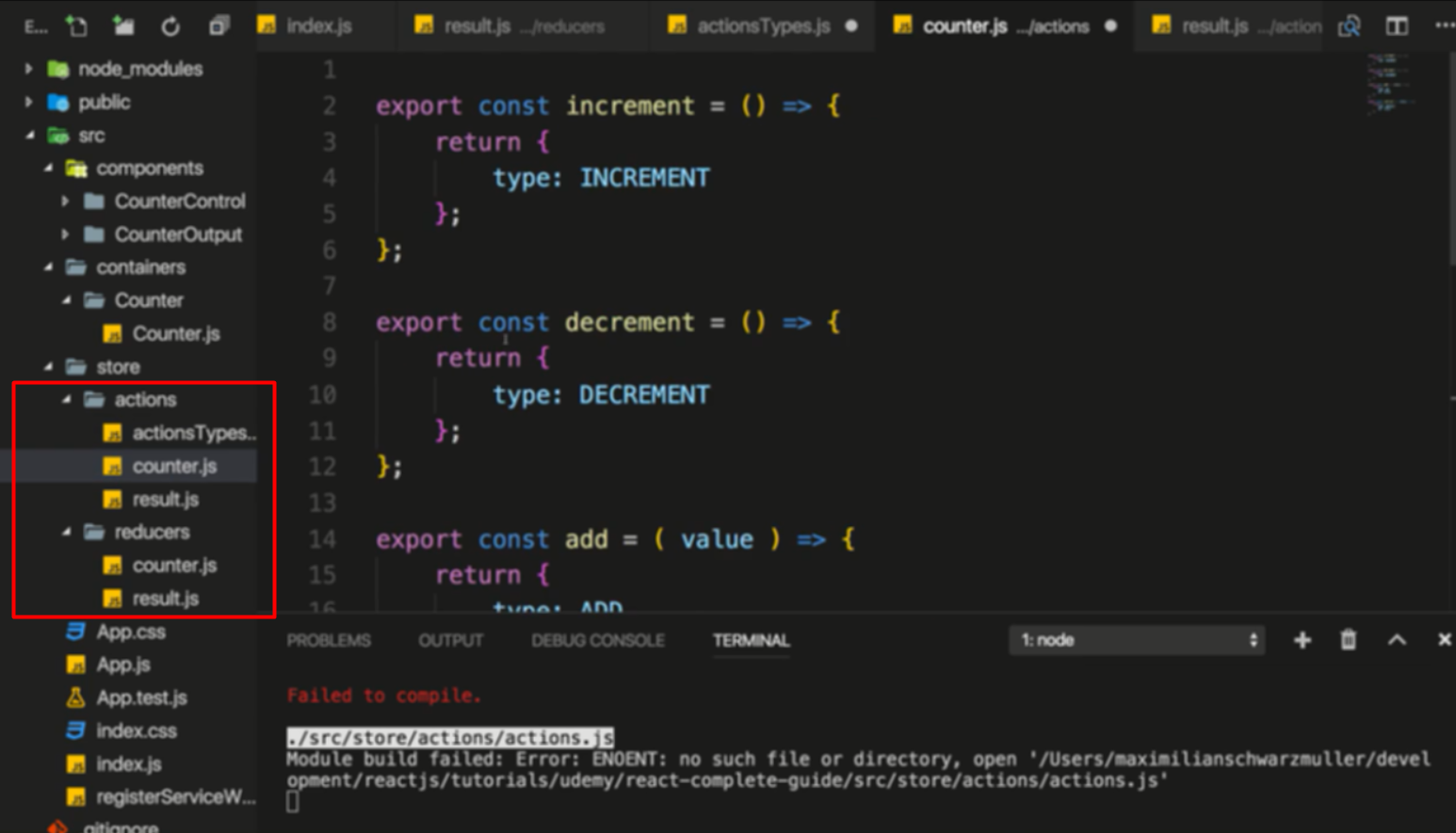
Quay lại khóa này



Sau 2 dây thì thực hiện action saveRessult với biến res được truyền vào

Cái hay ở đây là trong 1 cái action, sau khi code không đồng bộ chạy xong gọi action khác

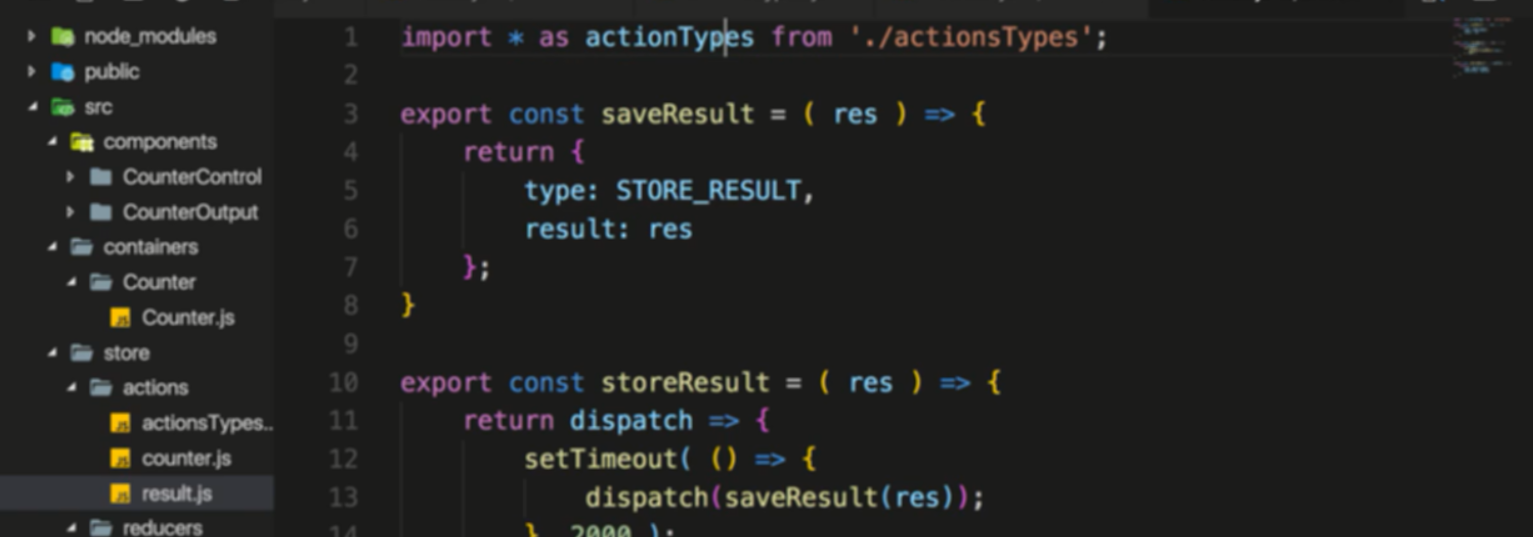
action saveResult lưu giá trị của response trong async vào state



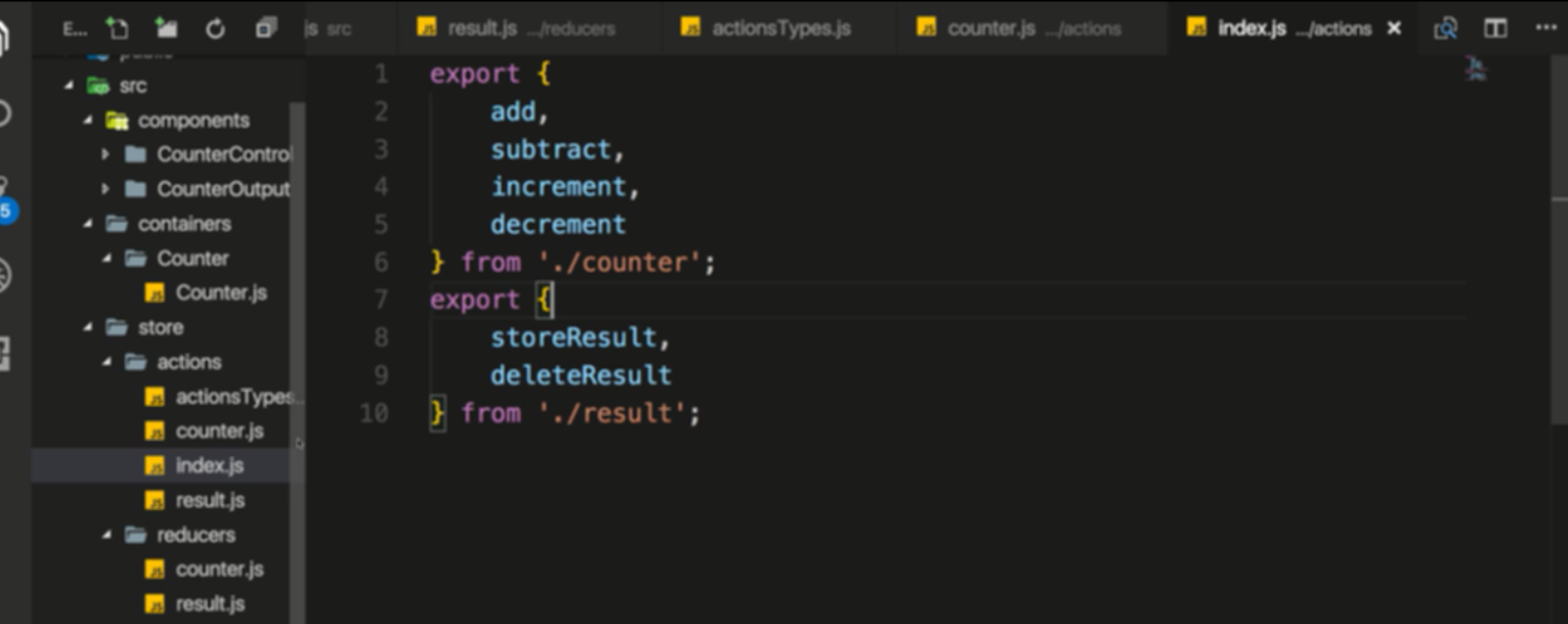
Tạo thư mục action chứa các action

action cũng chia ra như reducer

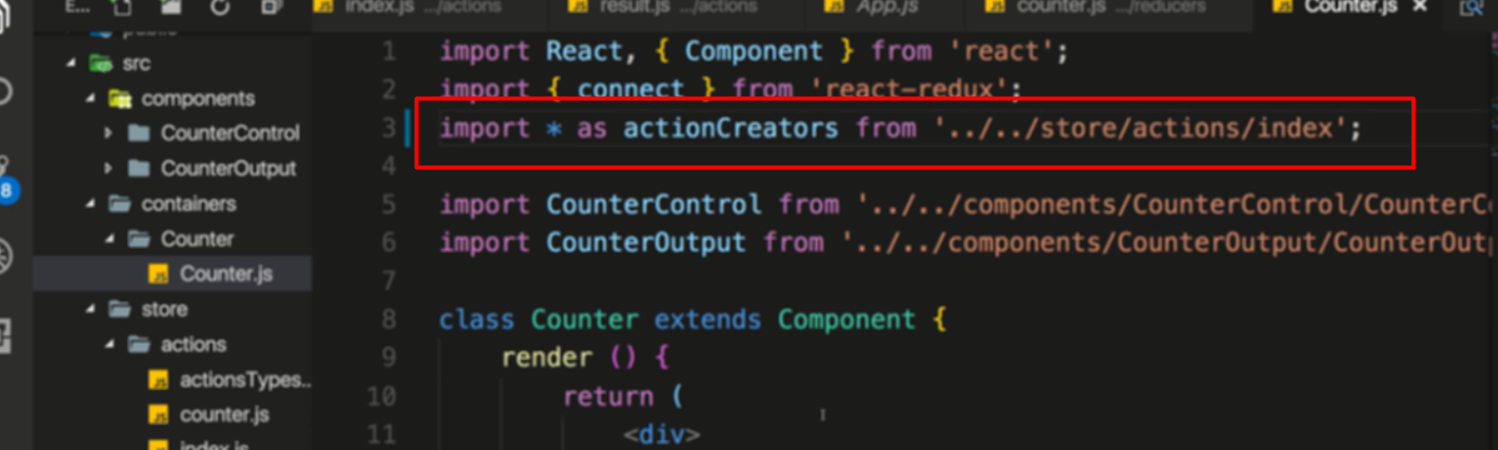
actionsType chưa các tên của action



Tất nhiên là vẫn phải import action type vào trong các action



Tạo file index.js trong action để quả lý export các action trong nhiều module trong 1 file

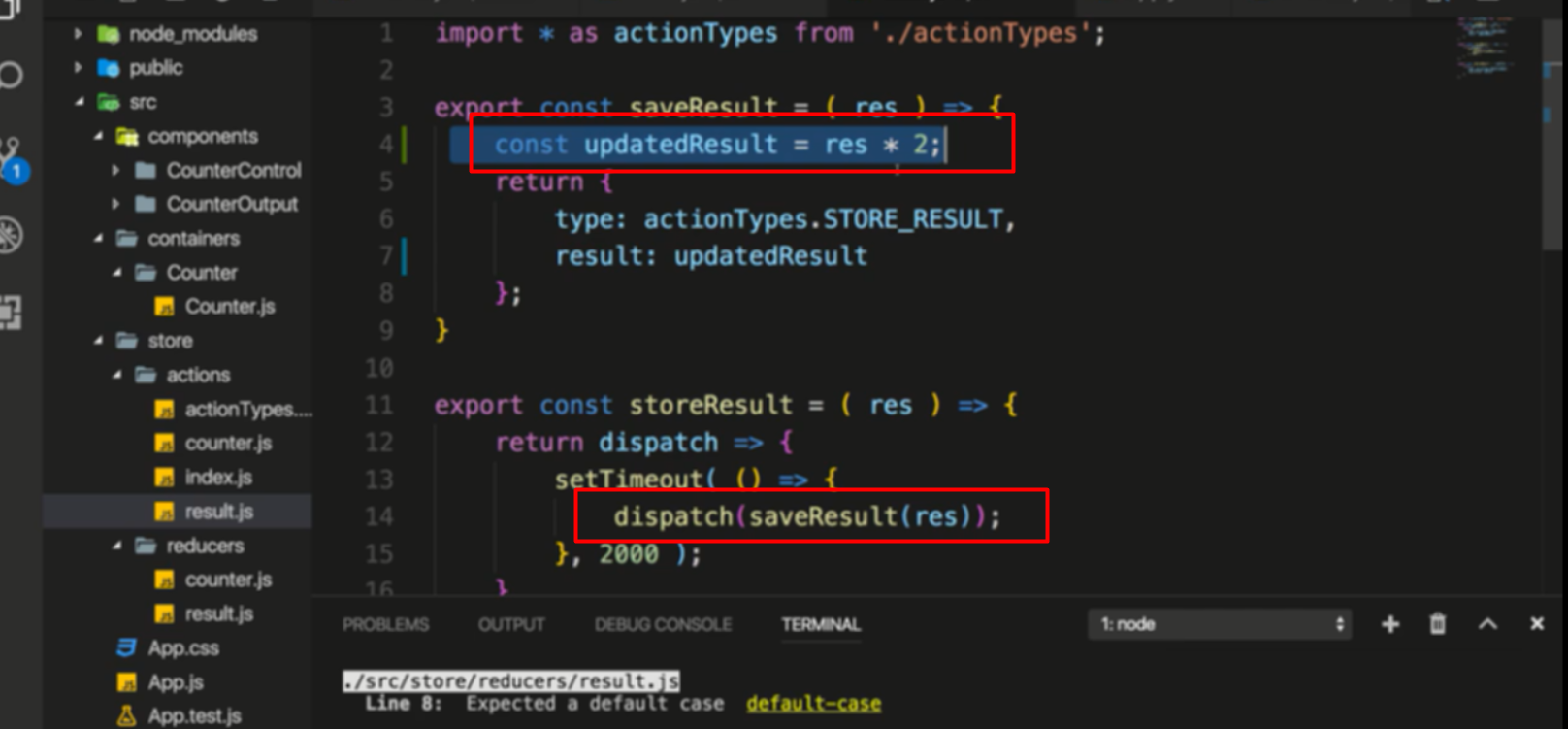


Tên được giữ nguyên nên ko cần đổi tên

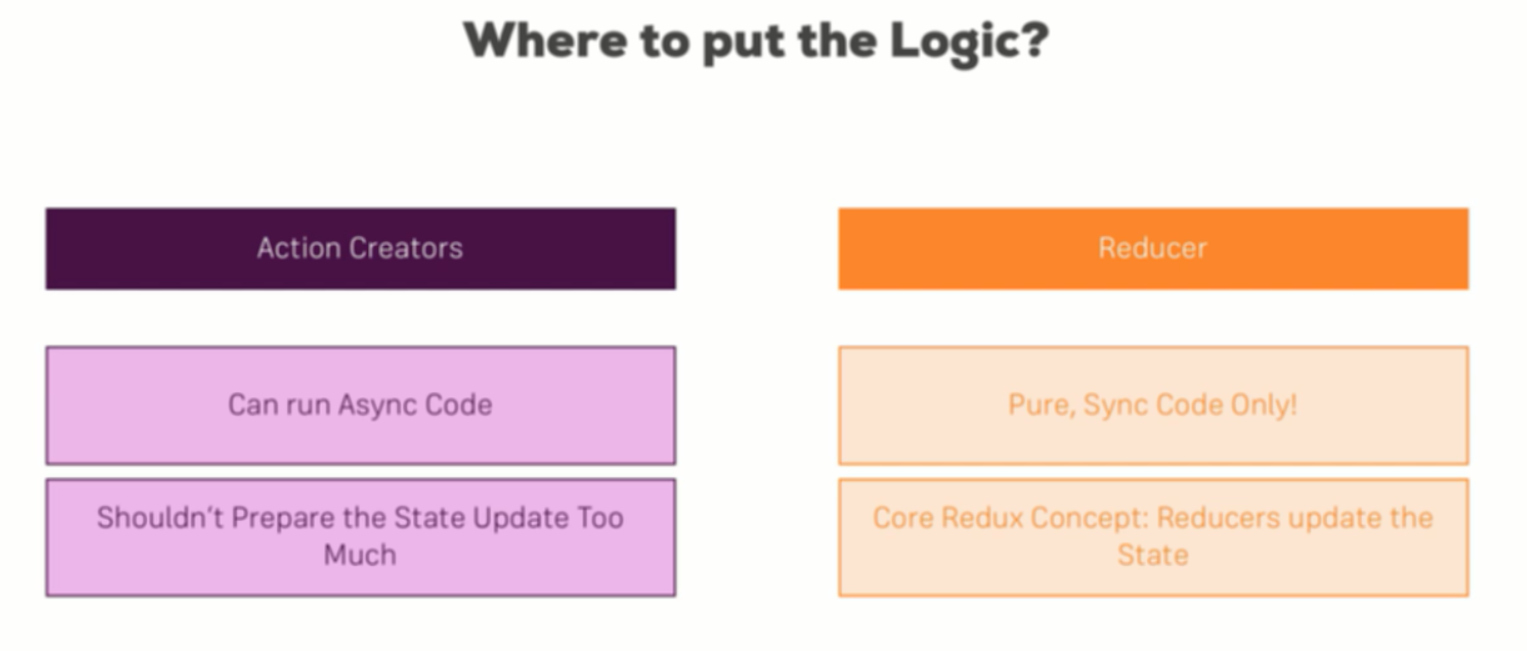
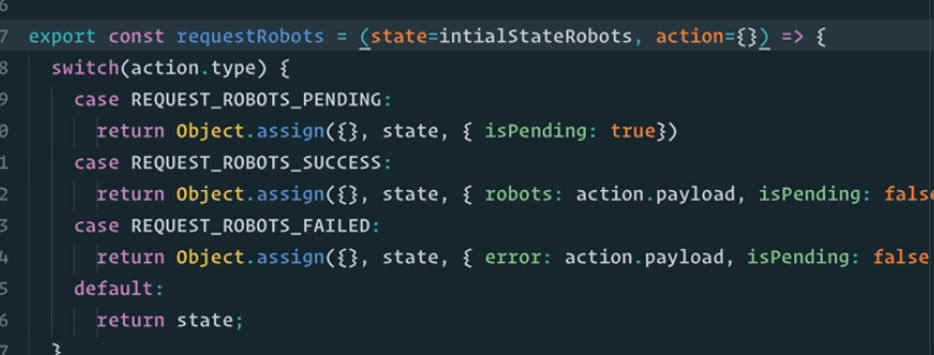
9. Where to Put Data Transforming Logic

Giá trị sau khi request async về nên set ở đâu

Có thể đưa giá trị đó vào ở action rồi reducer lấy action đó xử lý

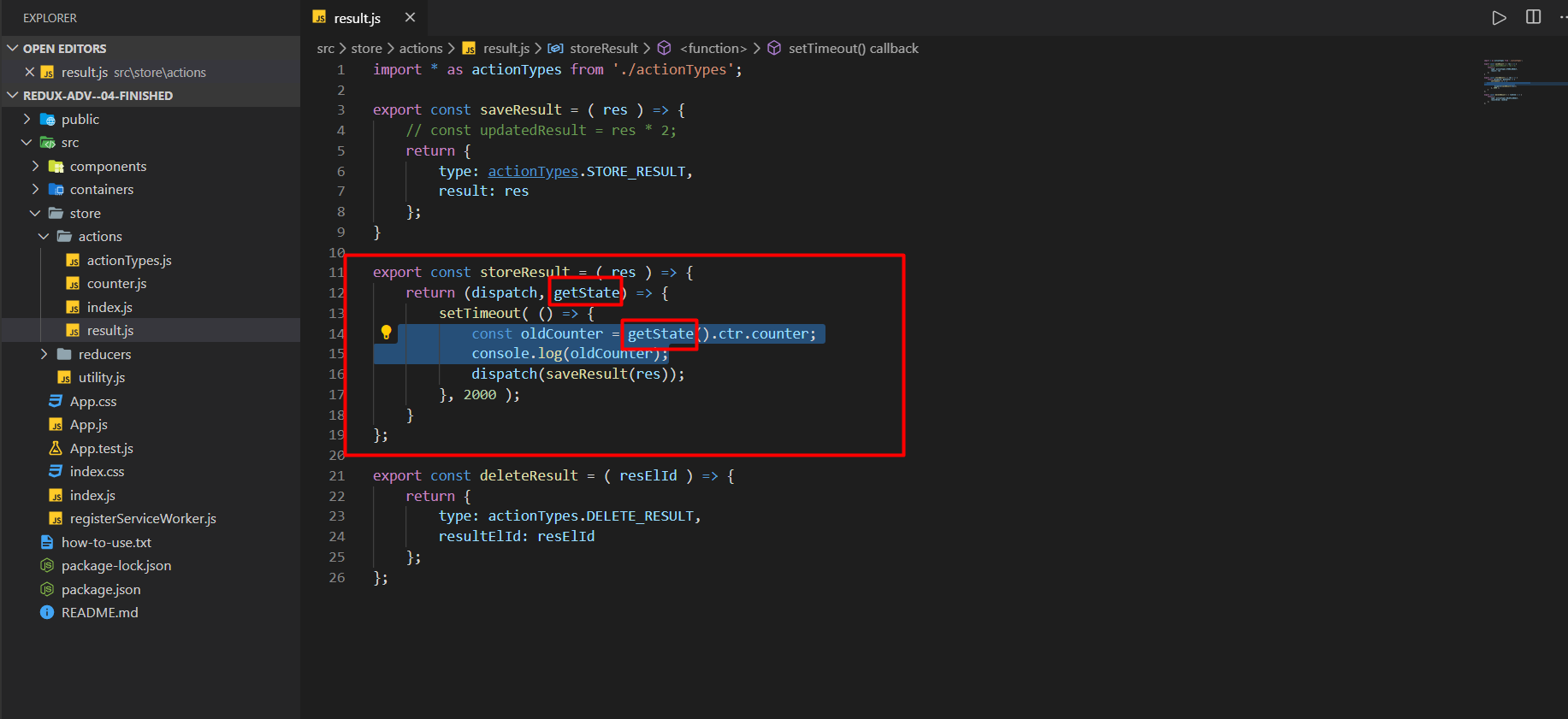


Hoặc như khóa zero to mastery, xử lý và set to state ở trong reducer



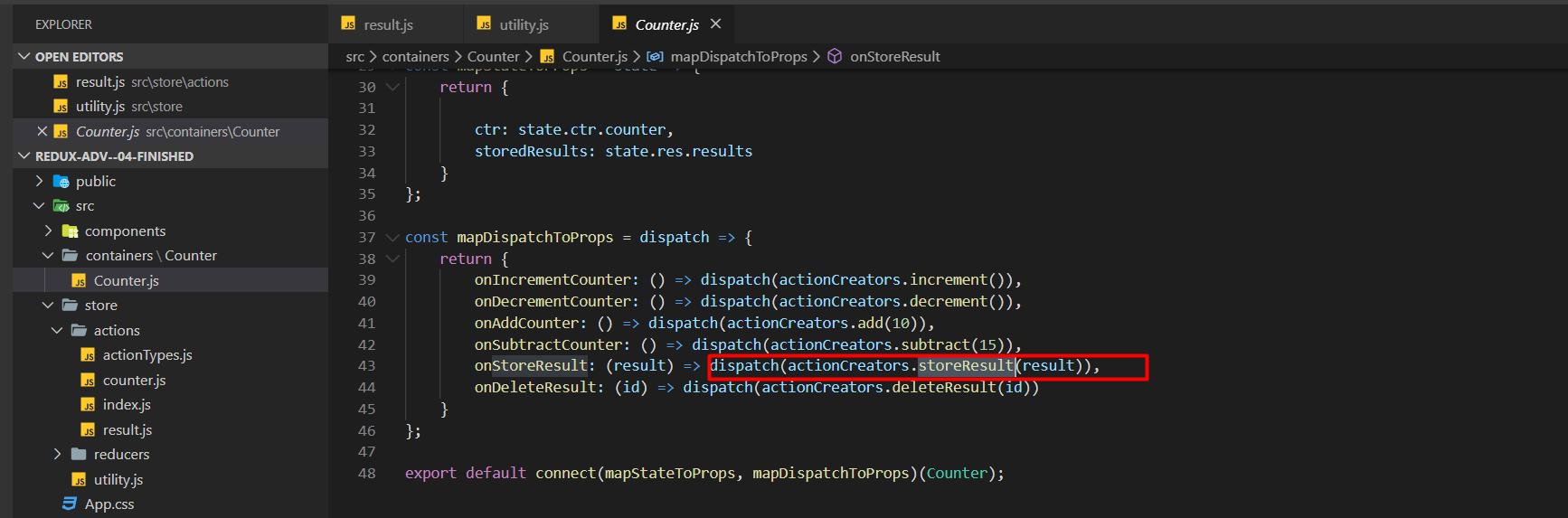
* Nên xử lý trong reducer như trong zero to master

10. Using Action Creators and Get State



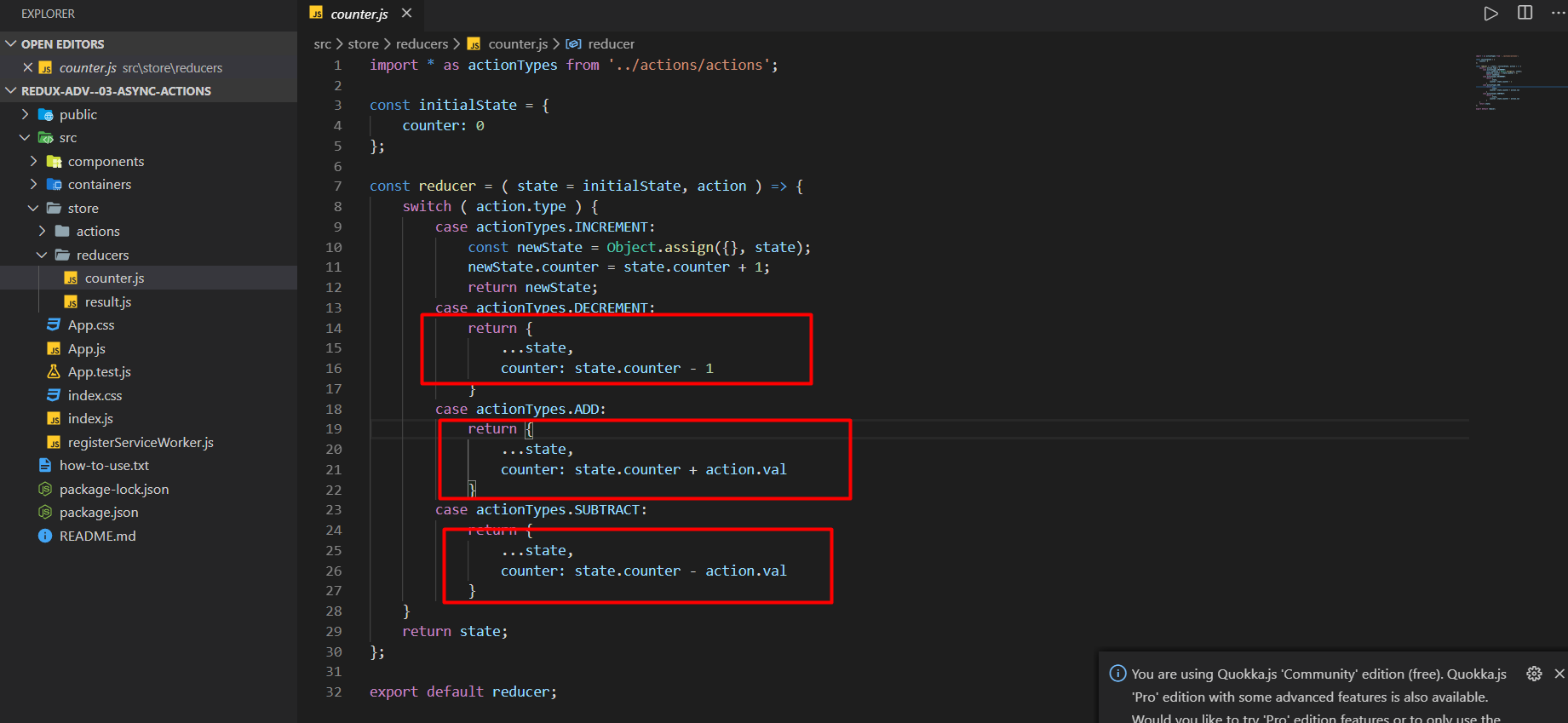
Hàm getstate lấy state tổng của redux

Nhưng thực ra nên truyền biến vào action storeResult khi gọi nó trên component thì tốt hơn, khi đó lúc định nghĩa action phải có biến đầu vào

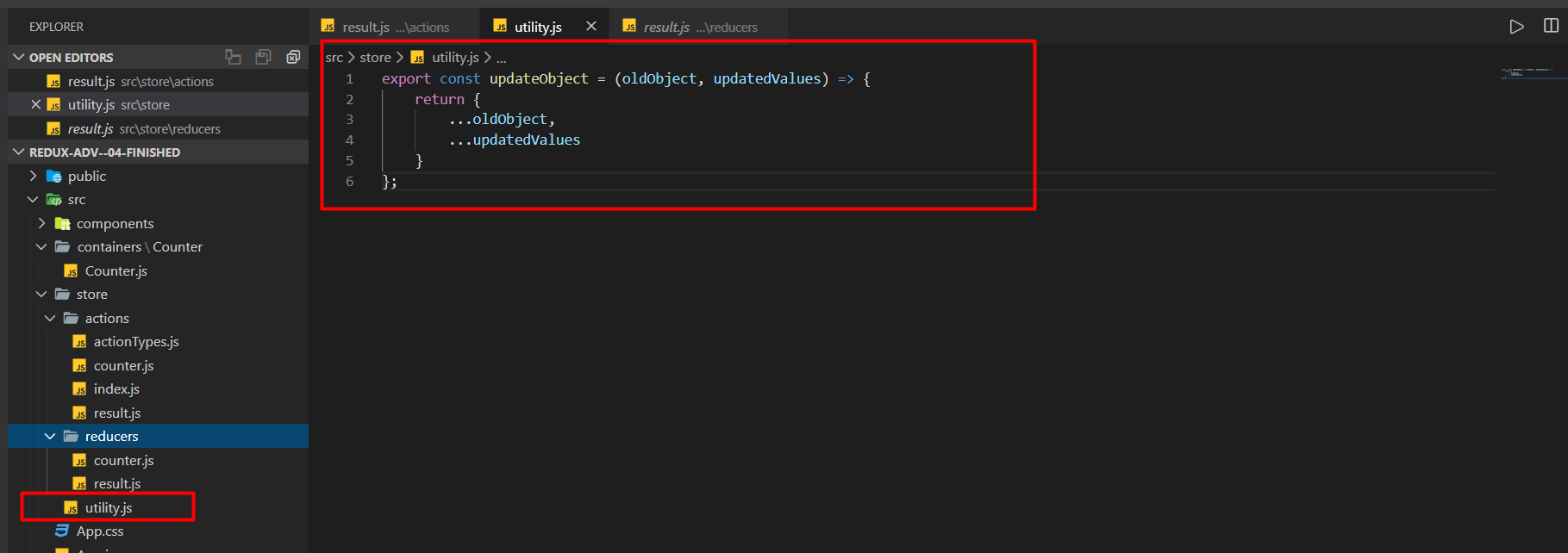


Có thể đưa biến vào ở đây

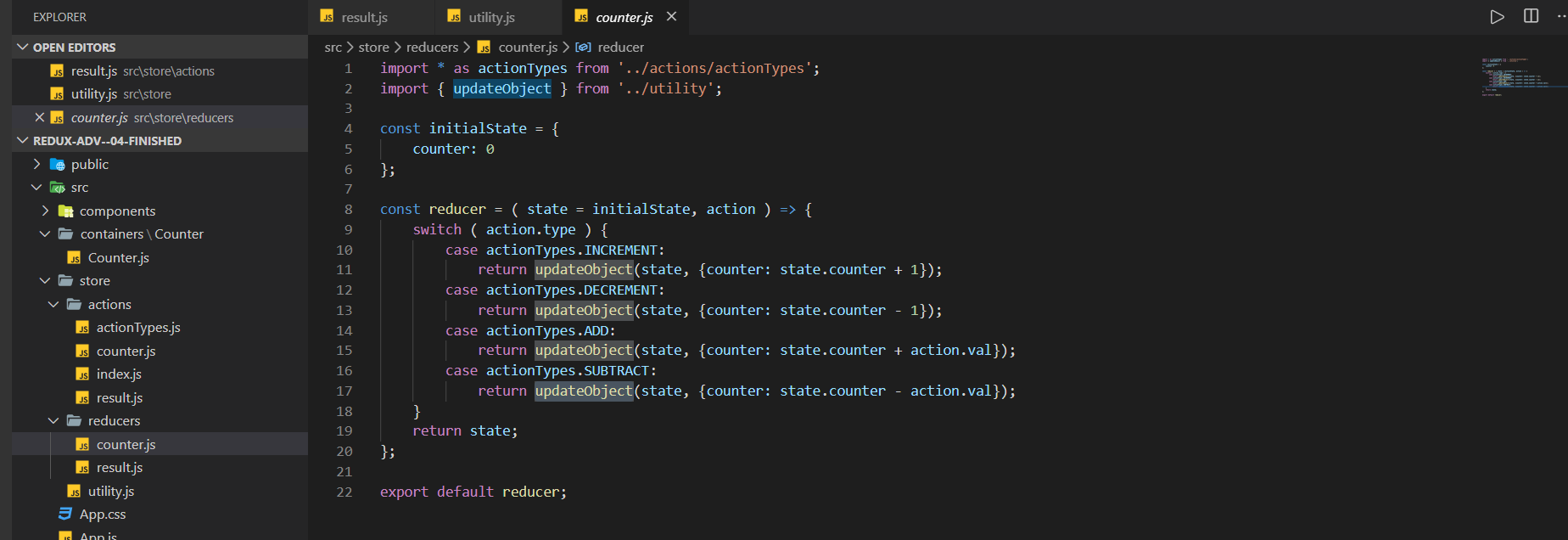
11. Using Utility Functions



Thực hiện mấy chổ return ni một cách clean code hơn

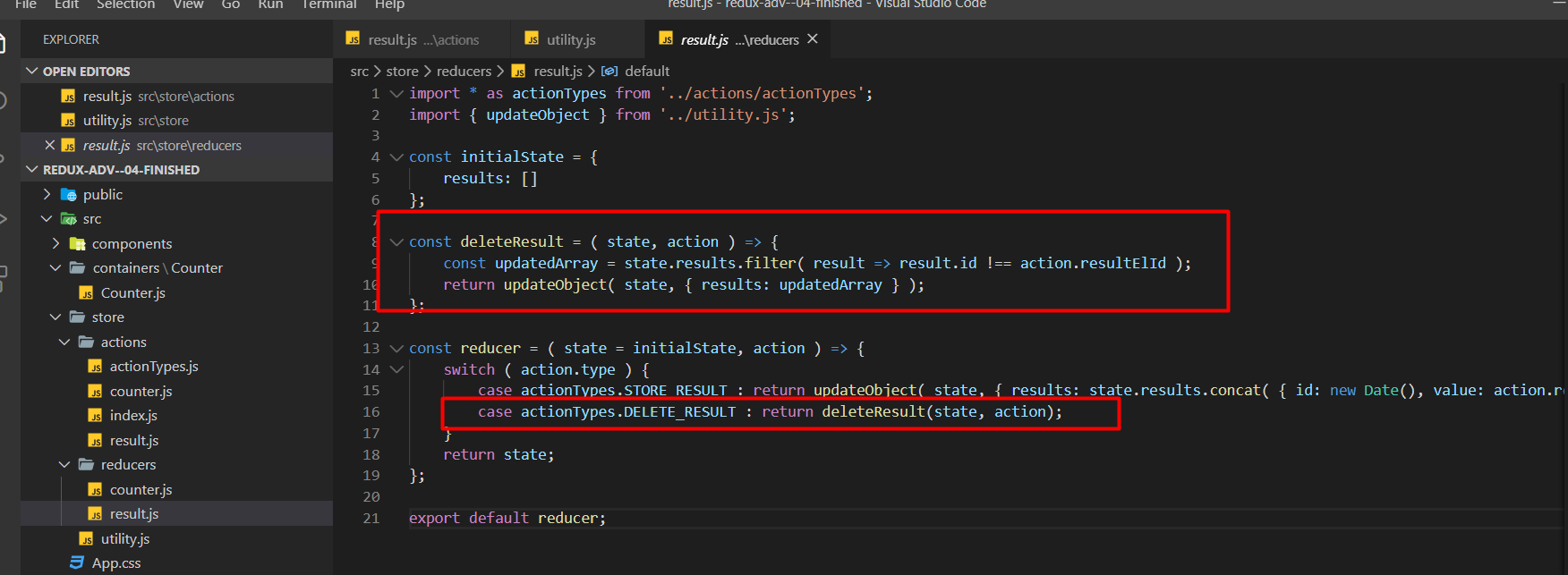


Tạo ultility



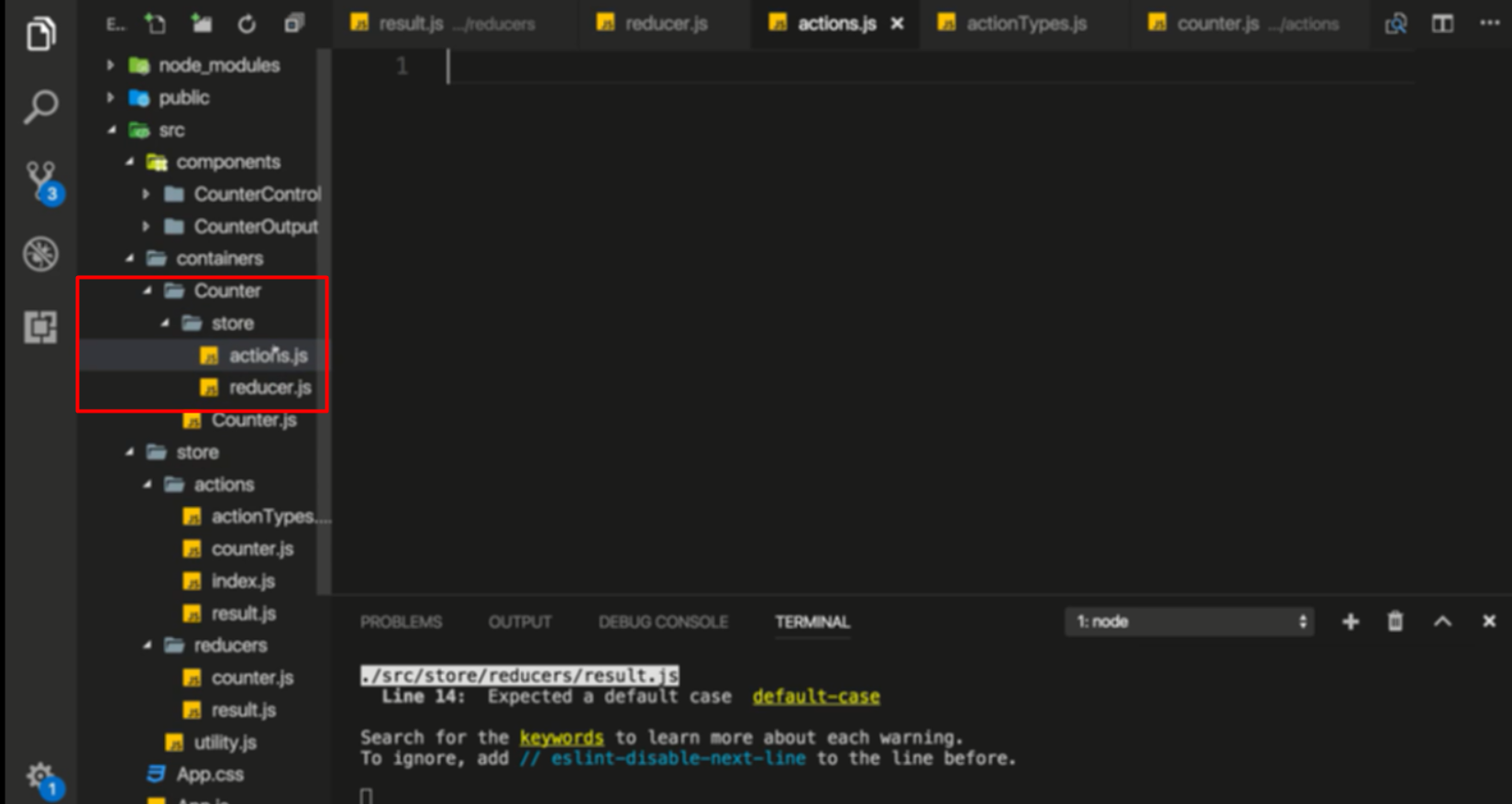
Import vào và sử dụng trong reducer

12. A Leaner Switch Case Statement



Đưa code trong case vào hàm để code gọn gàng hơn

13. An Alternative Folder Structure



Có thể tổ chức folder bằng cách tạo thư mục store cho từng cotainer để handler redux cho container đó