Contents

[Setting Boiler Plate for Redux 2](#_Toc17224138)

[Step 1 [Install dependencies: 2](#_Toc17224139)

[Step 2: [Create Store {createStore, initializeState, compose, applyMiddleWare}] 2](#_Toc17224140)

[Step 3: [Import Provider (from ‘react-redux’) and Store in App.js] 3](#_Toc17224141)

[Step 4: [Wrap Everything in App.js in Provider and Pass Store [<Provider store={store}>]] 3](#_Toc17224142)

[Step 5: [Create Reducers Folder and the Corresponding index.js file for ‘combineReducers’ (from redux)and add user Defined Reducer as well {contactReducer}] 4](#_Toc17224143)

[Step 6: [Create contactReducer (Custom Reducer)] 4](#_Toc17224144)

[Connecting Data to Components and Dispatching Actions: 5](#_Toc17224145)

[Step 1: [Define Types in Action Folder] 5](#_Toc17224146)

[Step 2: [Copy State in contactReducer, import action from Types.js and use it] 6](#_Toc17224147)

[Step 3: [Calling the GET\_CONTACTS action from Contacts.js Component] [This is the Component that is supposed to display ‘Contacts’] 7](#_Toc17224148)

[Sub step 1 [Import {Connect} from ‘react-redux’] 7](#_Toc17224149)

[Sub Step 2: [Import GET\_CONTACTS (Action Type) from types] 8](#_Toc17224150)

[Sub Step 3: [Create mapStateToProps function, This will Get State from contactReducer (Appropriate Reducer) to the Component] 9](#_Toc17224151)

[Sub Step 4: [Create mapDispatchToProps function, This will be calling the appropriate action type {GET\_CONTACTS}, this action type, will return updated contents of state related to appropriate reducer { contactReducer}, this update state, will be equivalent to “state”, which we have parameterized to “mapStateToProps” function ] 10](#_Toc17224152)

[Sub Step 5 : [Export “mapStateToProps” & “mapDispatchToProps” using Connect in the following format] 10](#_Toc17224153)

[Sub Step 6: [Finally, we have to call getContactsForCurrentComponent in the “componentDidMount” to ensure for data to be available as the component loads] 11](#_Toc17224154)

[Sub Step 7: Finally the data would be available in **variable** highlighted below, data will be available only after calling “getContactsForCurrentComponent()” function in “ComponentDidMount()” 12](#_Toc17224155)

[Sub Step 8: Moving the Function which calls the Actions to a Separate File 12](#_Toc17224156)

[Express Way for Learning of Actions: 14](#_Toc17224157)

[Step 1: [Add Required Action Labels to types.js] 14](#_Toc17224158)

[Step 2: [Create Functions in the required “<contact>Actions” file which manipulates Global State [Data]] 15](#_Toc17224159)

[Step 3: [Call these Functions in the which calls Actions ] 15](#_Toc17224160)

[Step 4: [From Here As usual, Import Connect and Call the Functions which inturn call the actions] [DELETE CONTACT] 16](#_Toc17224161)

[Step 5: [From Here As usual, Import Connect and Call the Functions which inturn call the actions] [ADD CONTACT] 17](#_Toc17224162)

[Calling APIS 18](#_Toc17224163)

[Get API [All the API needs to be called in customAction.js file {contactActions.js}] 18](#_Toc17224164)

[Post: 19](#_Toc17224165)

[The Sequence: 20](#_Toc17224166)

[One 20](#_Toc17224167)

[Two: 21](#_Toc17224168)

[Three: 22](#_Toc17224169)

[Four: 23](#_Toc17224170)

[Five 24](#_Toc17224171)

[Final 25](#_Toc17224172)

Redux Tutorial:

# Setting Boiler Plate for Redux

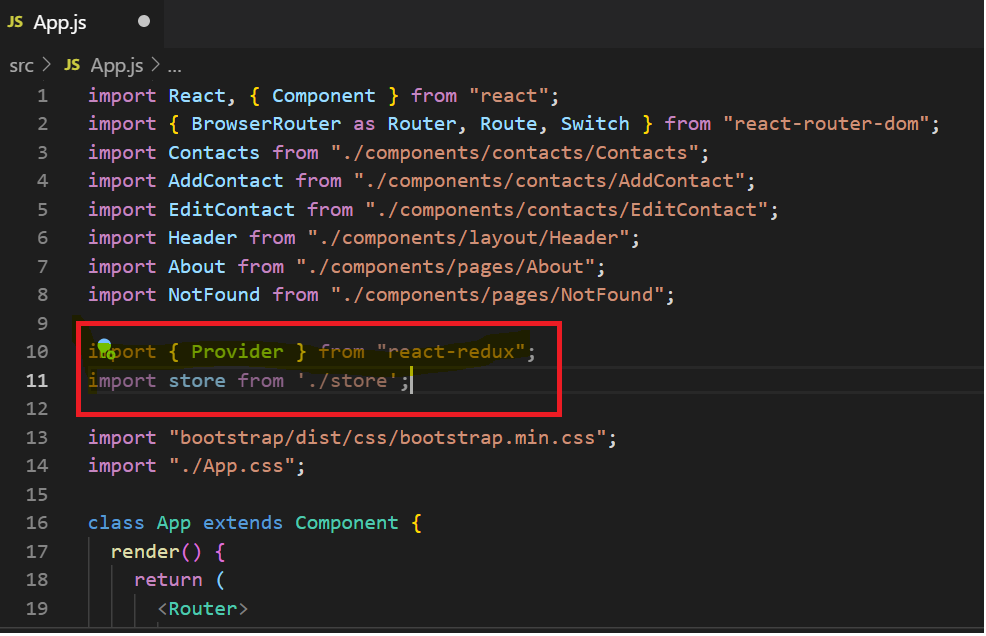
## Step 1 [Install dependencies:

npm i redux react-redux redux-thunk

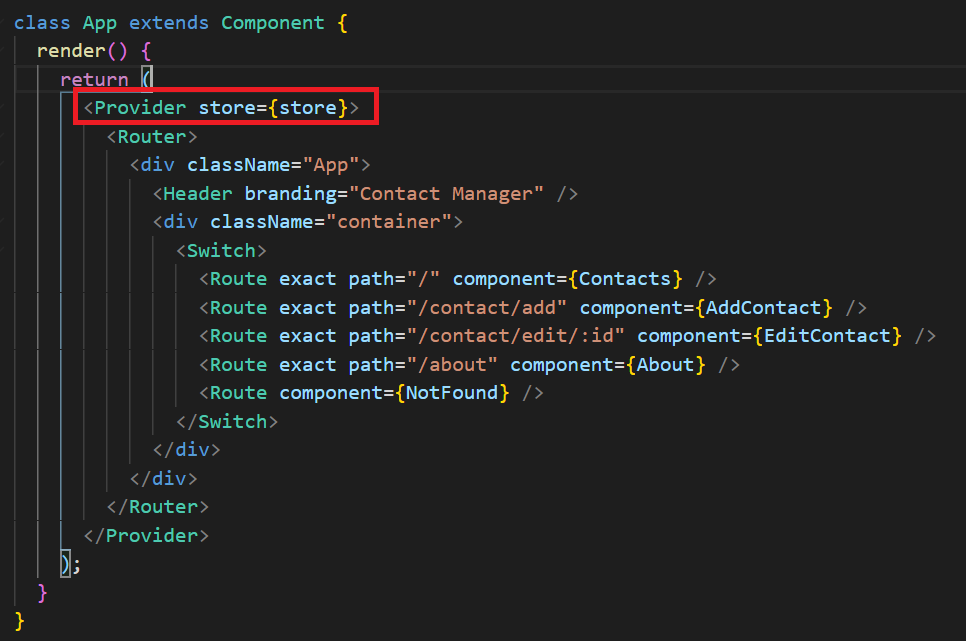
## Step 2: [Create Store {createStore, initializeState, compose, applyMiddleWare}]



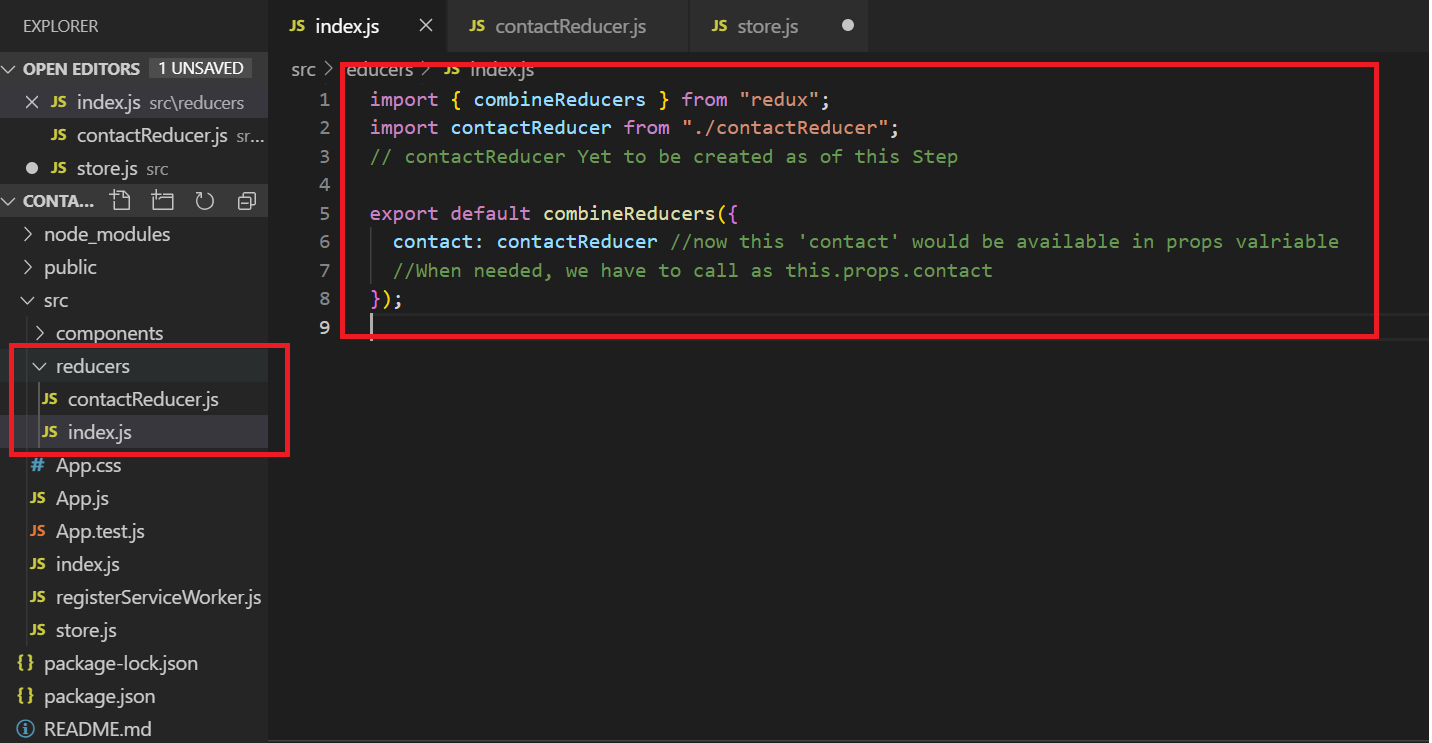
## Step 3: [Import Provider (from ‘react-redux’) and Store in App.js]



## Step 4: [Wrap Everything in App.js in Provider and Pass Store [<Provider store={store}>]]



## Step 5: [Create Reducers Folder and the Corresponding index.js file for ‘combineReducers’ (from redux)and add user Defined Reducer as well {contactReducer}]

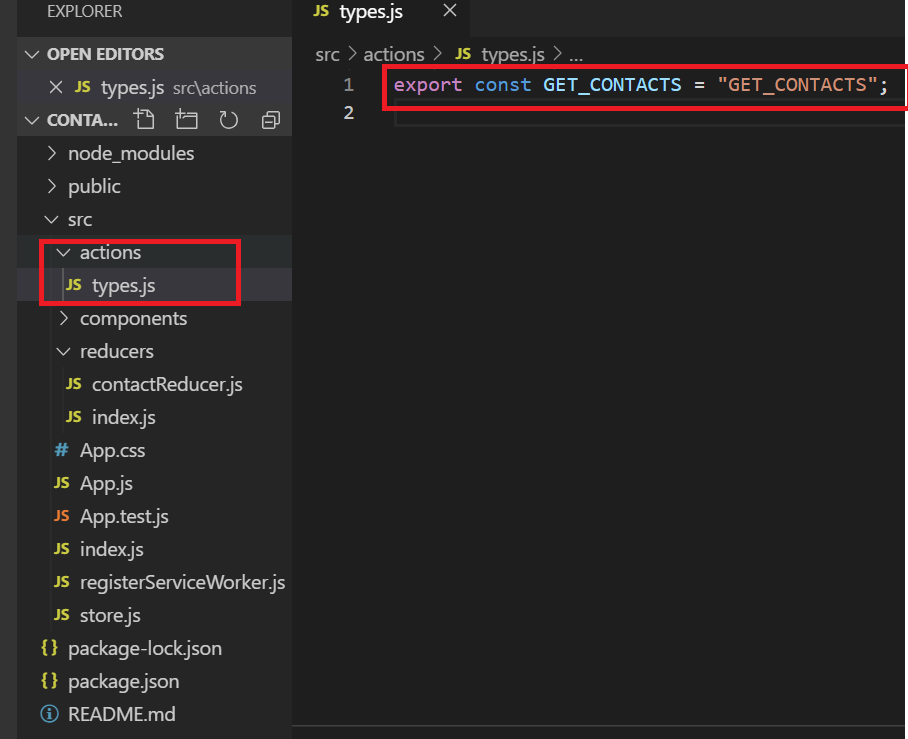


## Step 6: [Create contactReducer (Custom Reducer)]

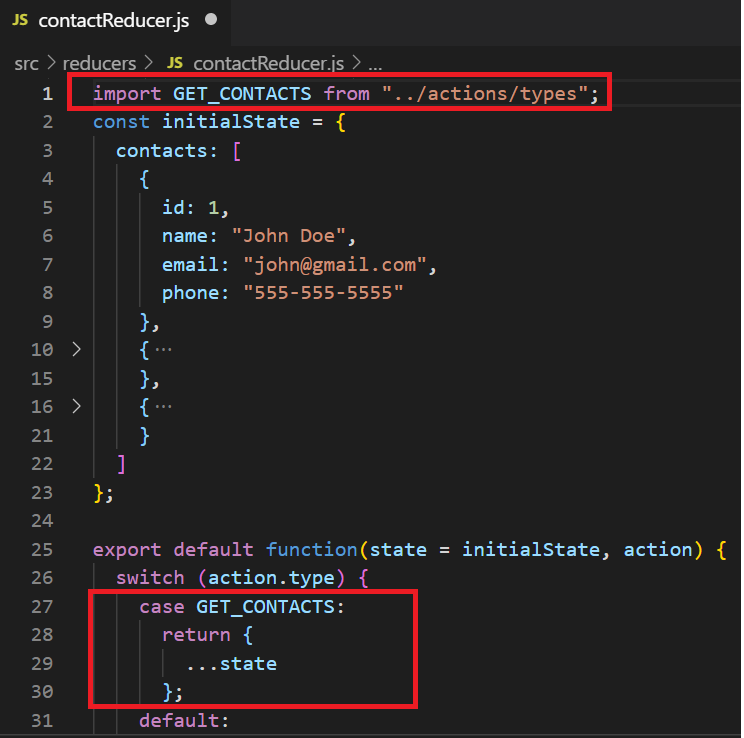


# Connecting Data to Components and Dispatching Actions:

## Step 1: [Define Types in Action Folder]



## Step 2: [Copy State in contactReducer, import action from Types.js and use it]

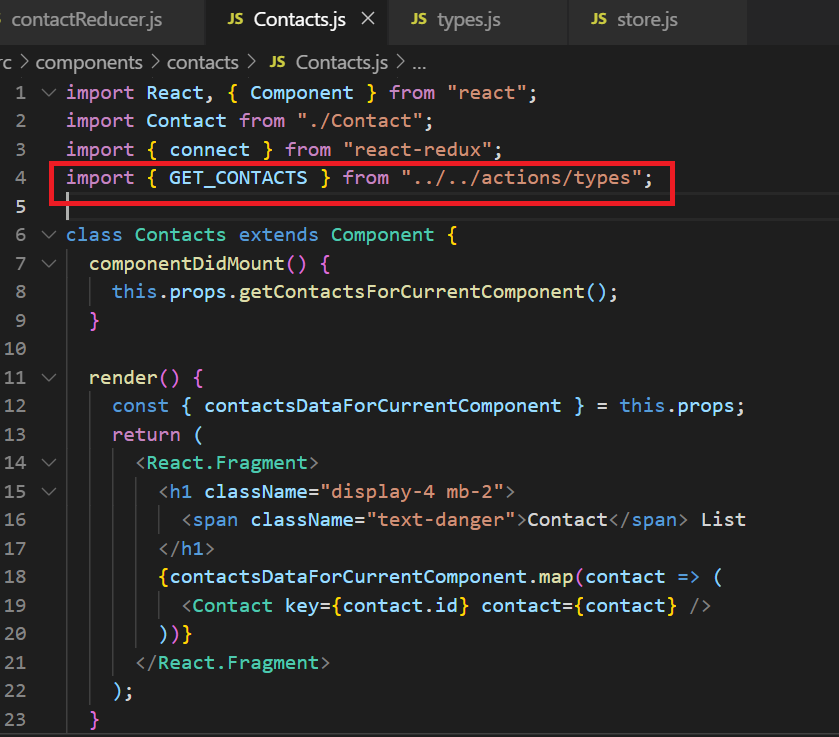


## Step 3: [Calling the GET\_CONTACTS action from Contacts.js Component] [This is the Component that is supposed to display ‘Contacts’]

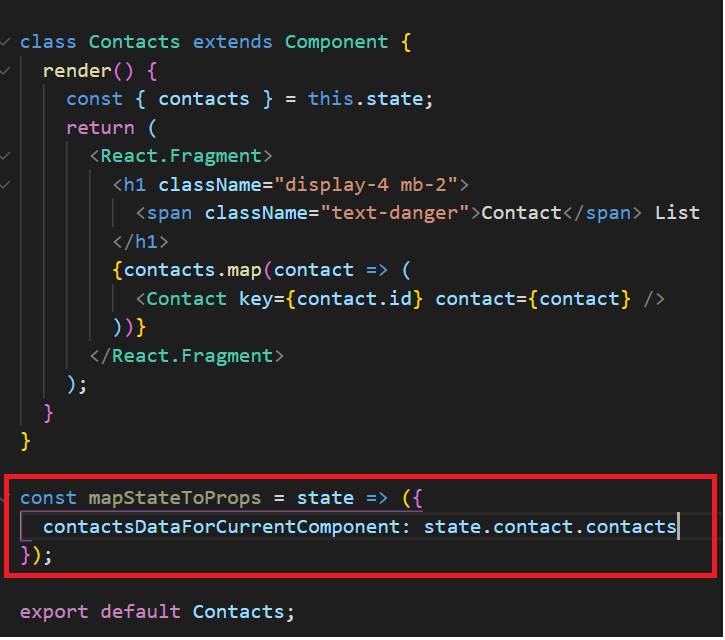
### Sub step 1 [Import {Connect} from ‘react-redux’]



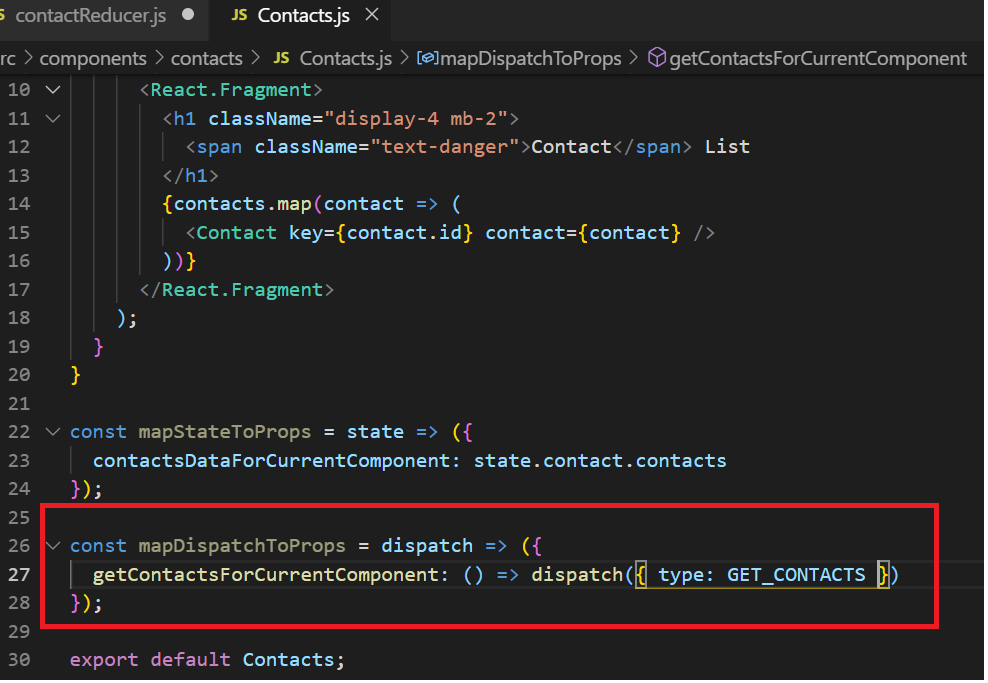
### Sub Step 2: [Import GET\_CONTACTS (Action Type) from types]



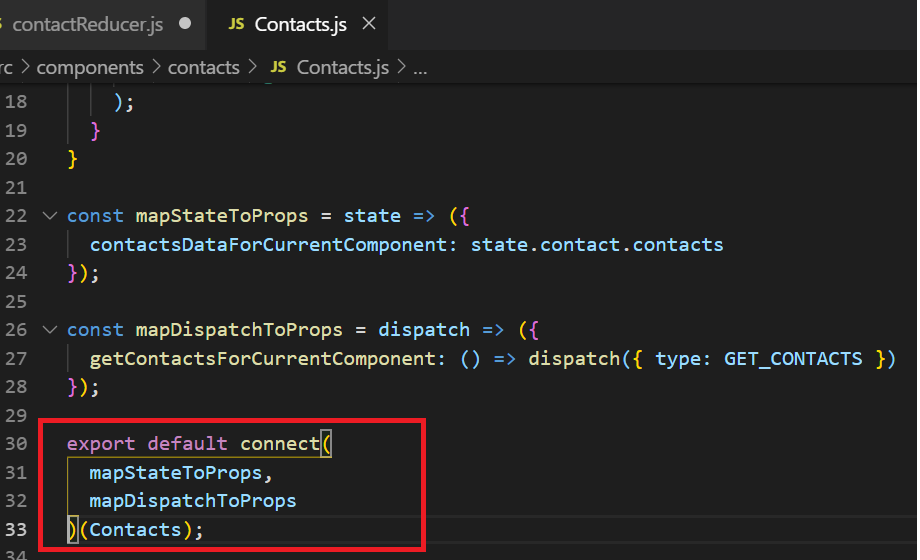
### Sub Step 3: [Create mapStateToProps function, This will Get State from contactReducer (Appropriate Reducer) to the Component]



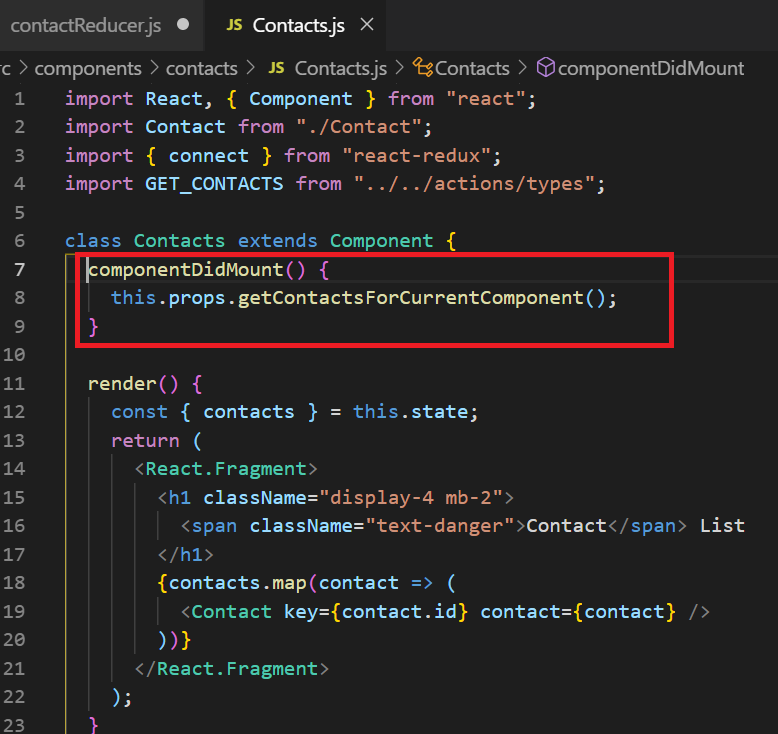
### Sub Step 4: [Create mapDispatchToProps function, This will be calling the appropriate action type {GET\_CONTACTS}, this action type, will return updated contents of state related to appropriate reducer { contactReducer}, this update state, will be equivalent to “state”, which we have parameterized to “mapStateToProps” function ]



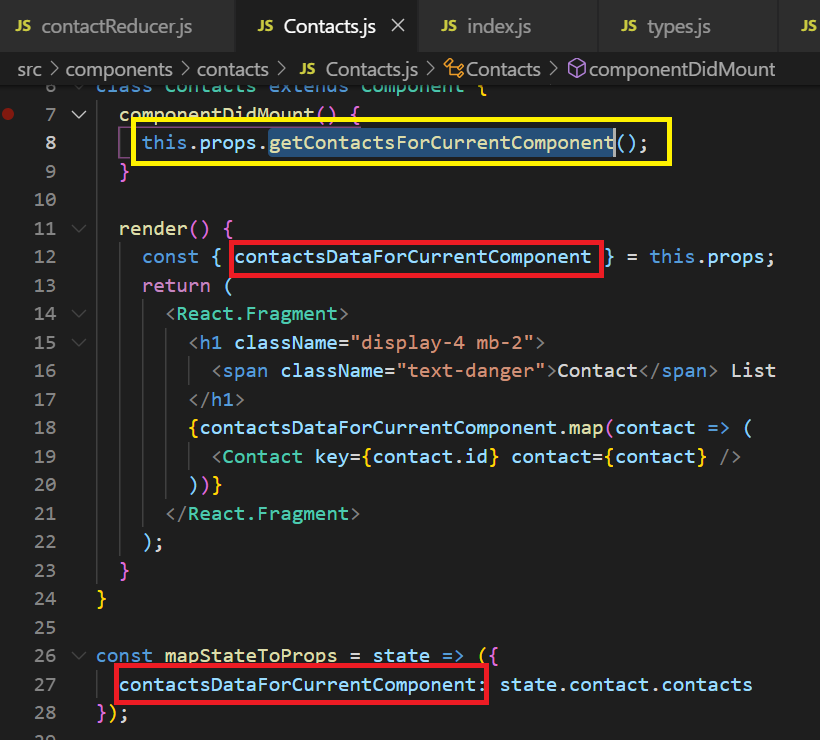
### Sub Step 5 : [Export “mapStateToProps” & “mapDispatchToProps” using Connect in the following format]



### Sub Step 6: [Finally, we have to call getContactsForCurrentComponent in the “componentDidMount” to ensure for data to be available as the component loads]

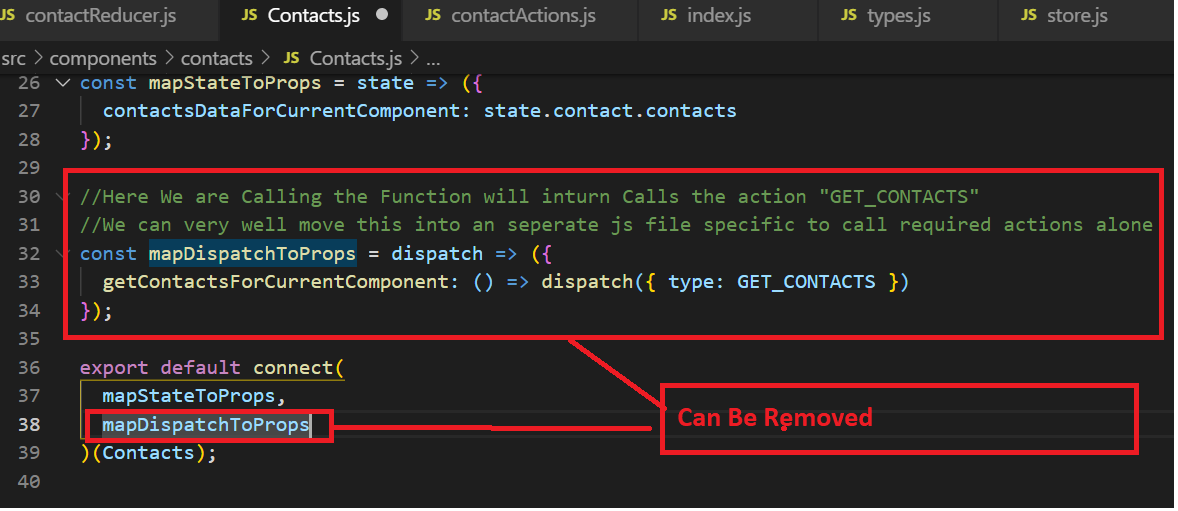


### Sub Step 7: Finally the data would be available in **variable** highlighted below, data will be available only after calling “getContactsForCurrentComponent()” function in “ComponentDidMount()”

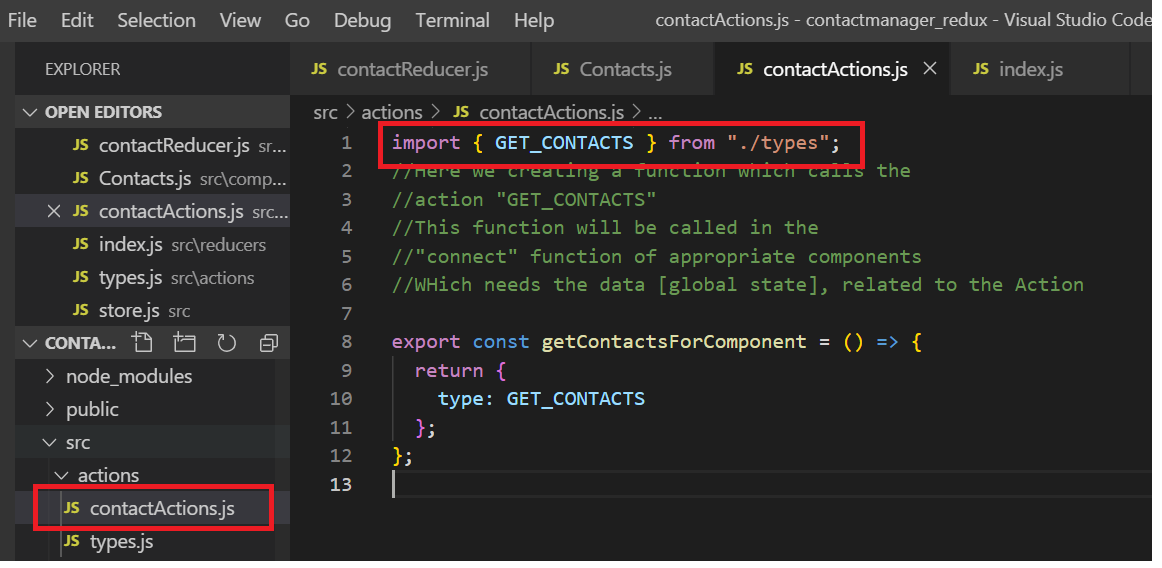


### Sub Step 8: Moving the Function which calls the Actions to a Separate File

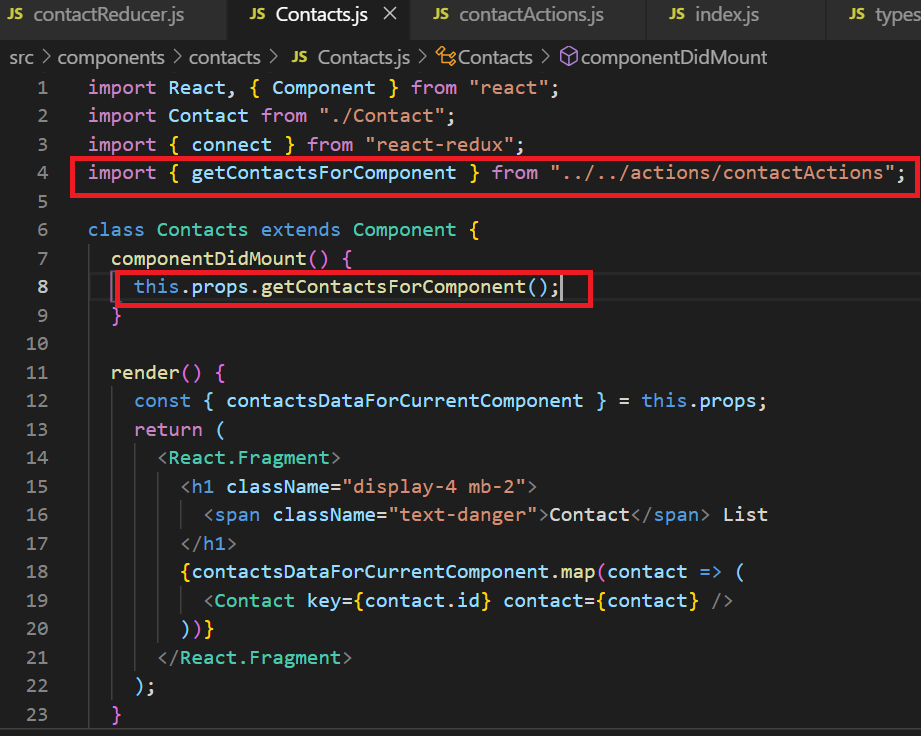
#### Super sub step1

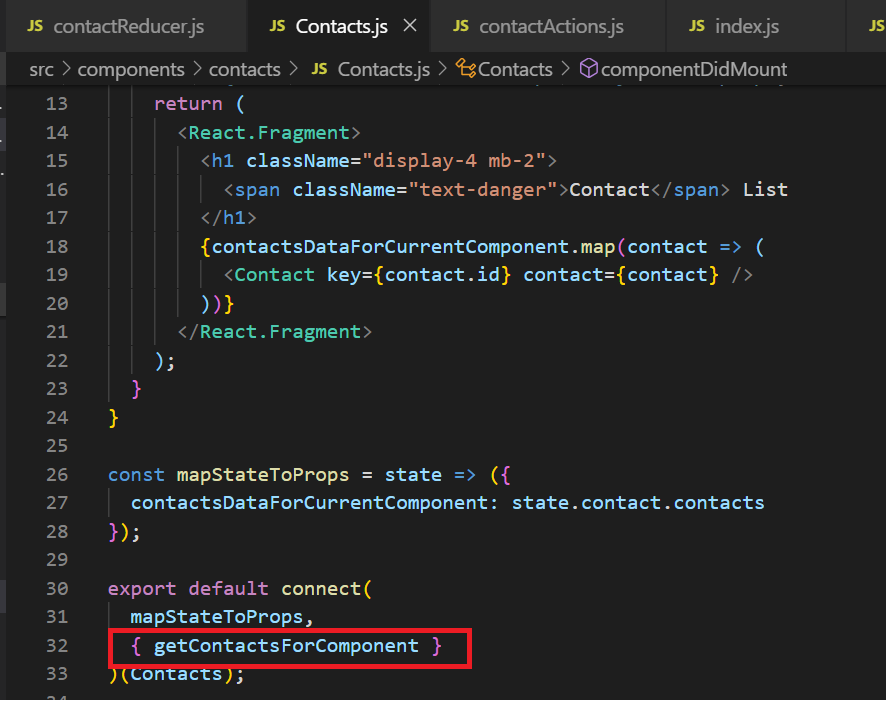


#### Super sub Step 2



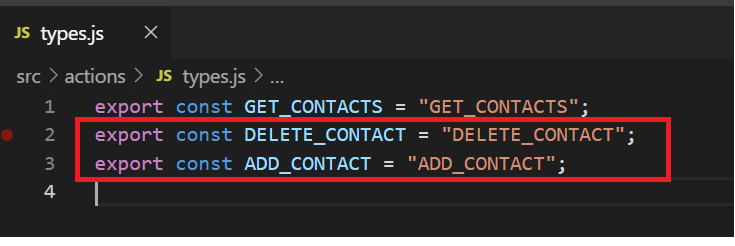
#### Super Sub Step 3:



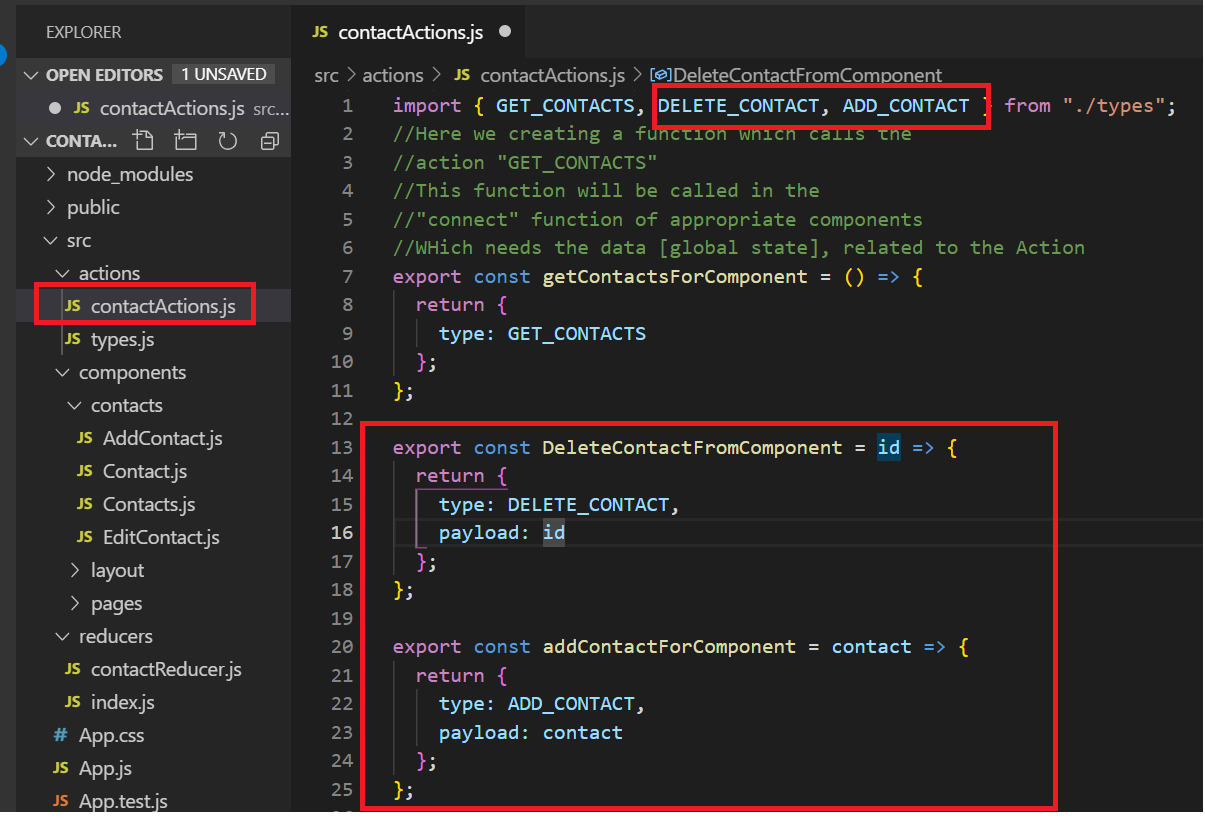


# Express Way for Learning of Actions:

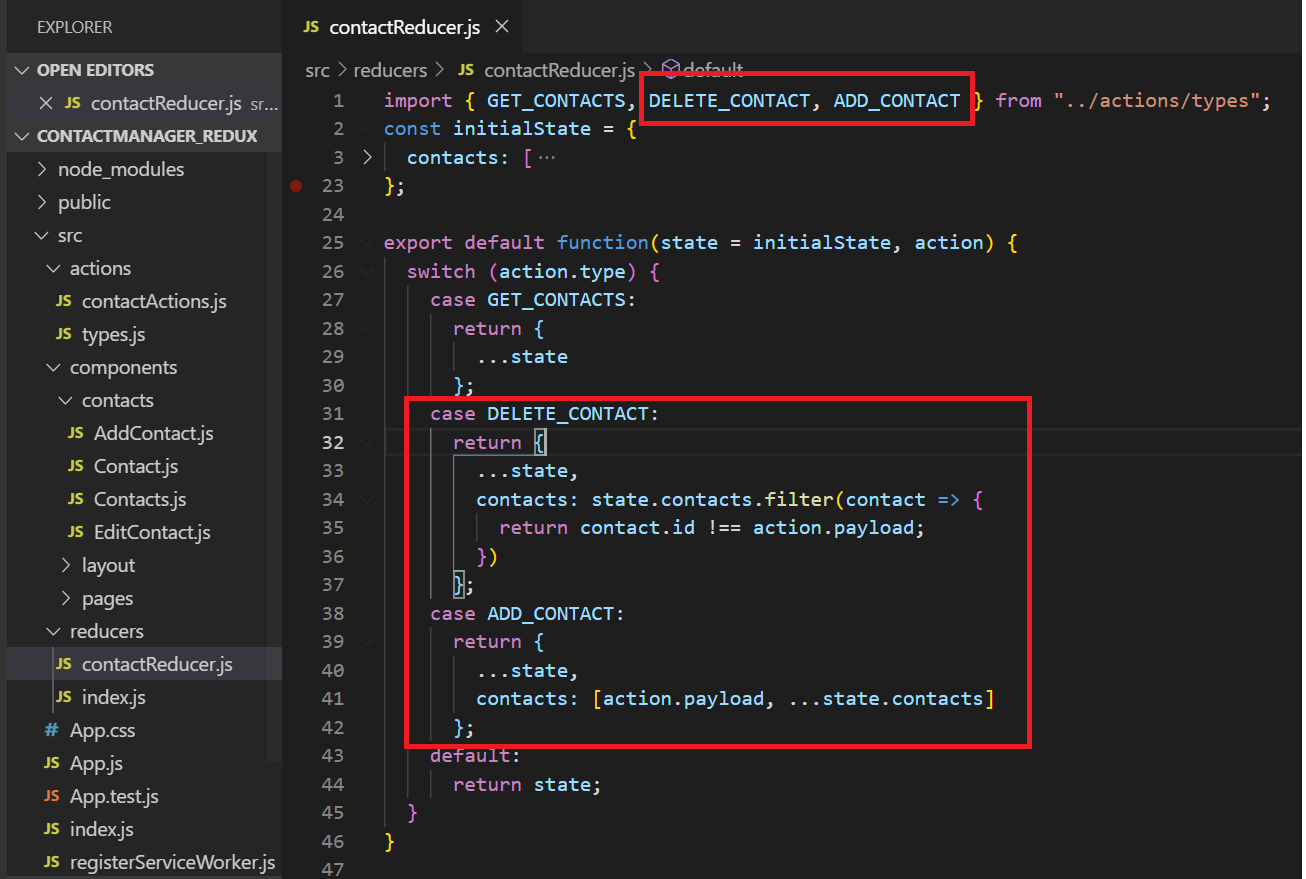
## Step 1: [Add Required Action Labels to types.js]



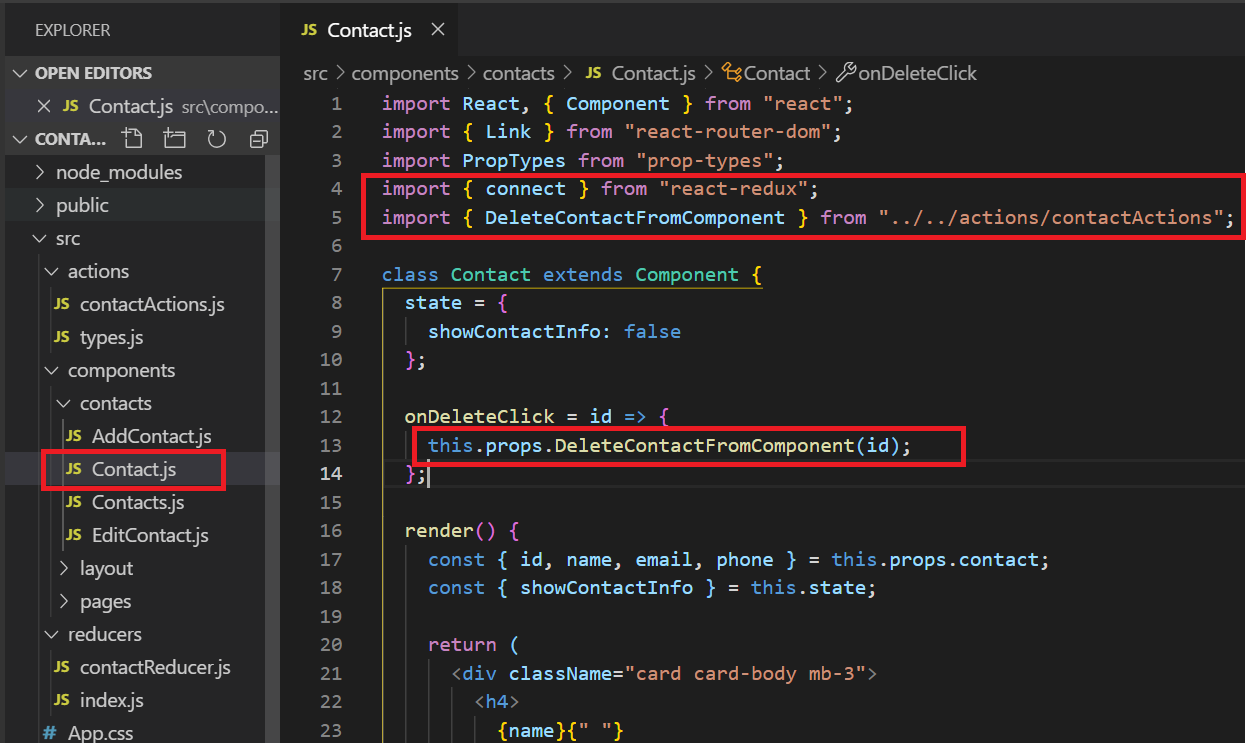
## Step 2: [Create Functions in the required “<contact>Actions” file which manipulates Global State [Data]]

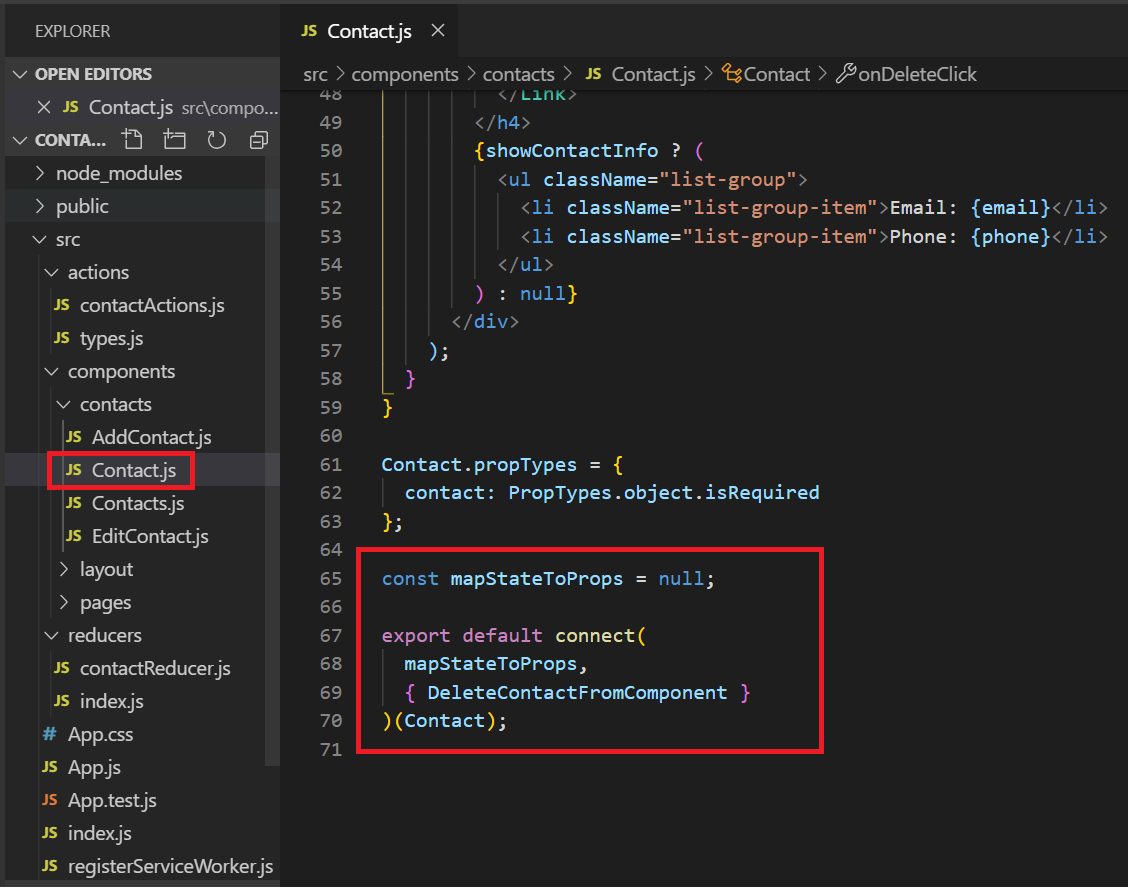


## Step 3: [Call these Functions in the which calls Actions ]

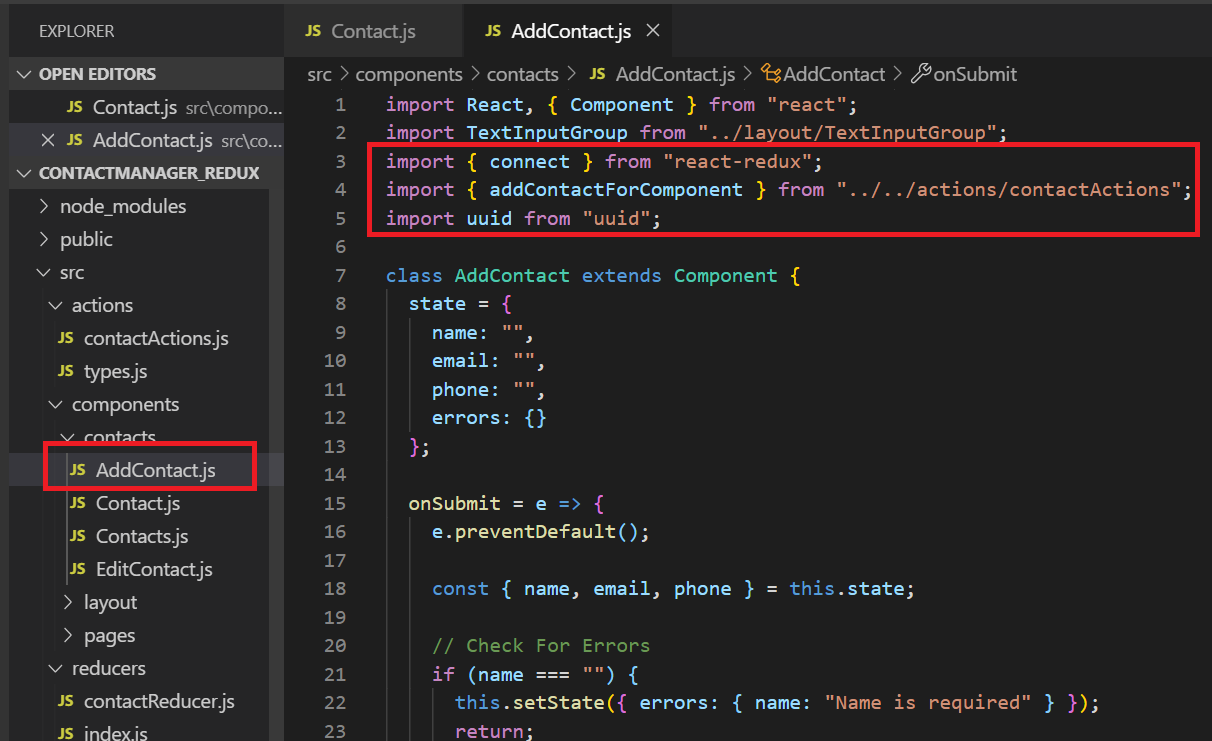


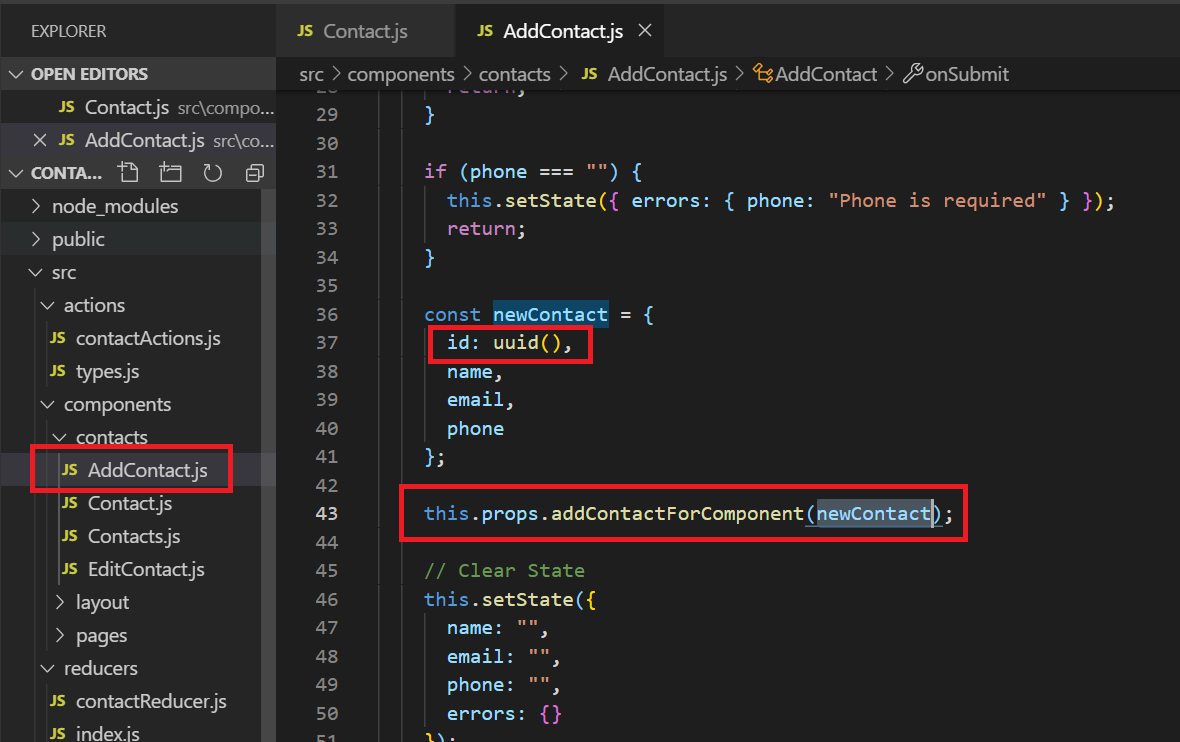
## Step 4: [From Here As usual, Import Connect and Call the Functions which inturn call the actions] [DELETE CONTACT]

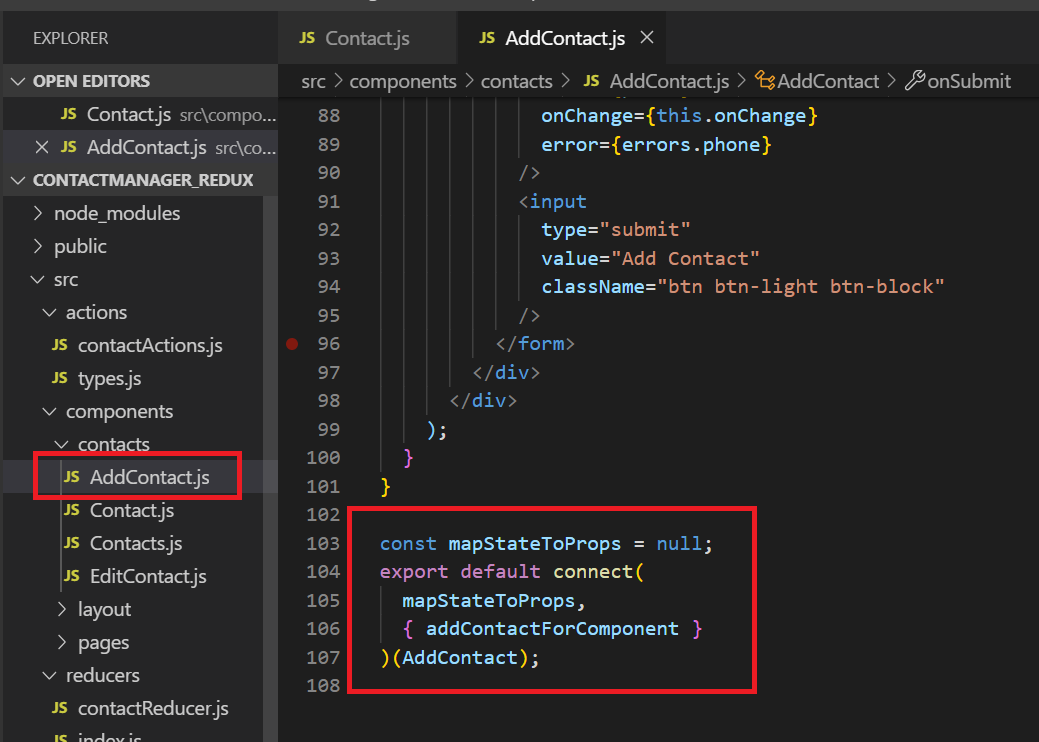




## Step 5: [From Here As usual, Import Connect and Call the Functions which inturn call the actions] [ADD CONTACT]

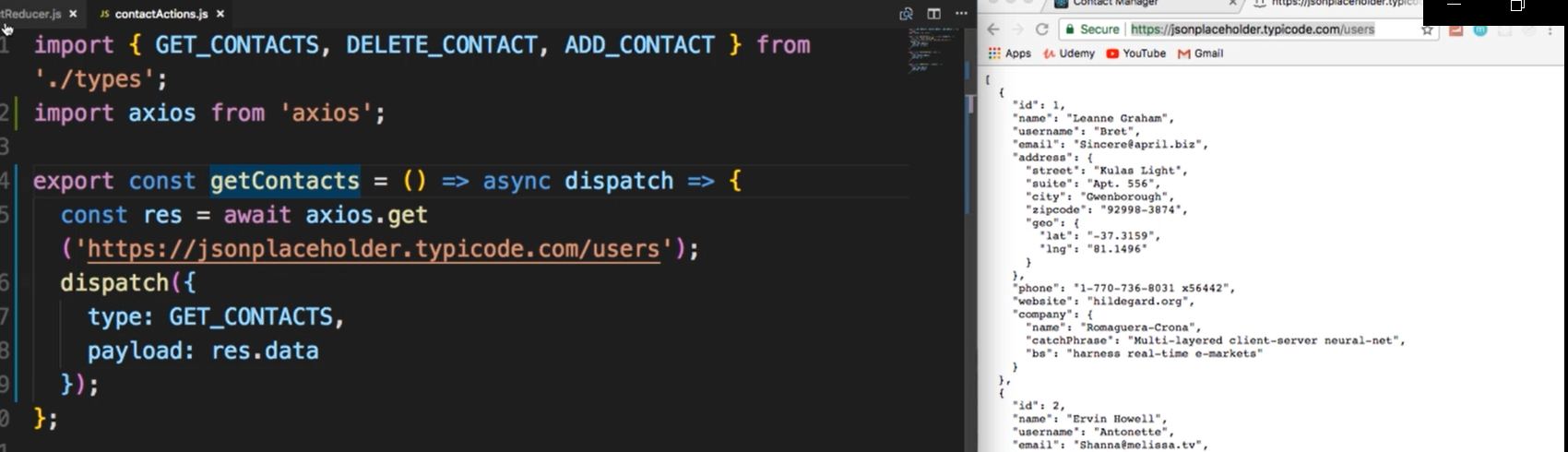


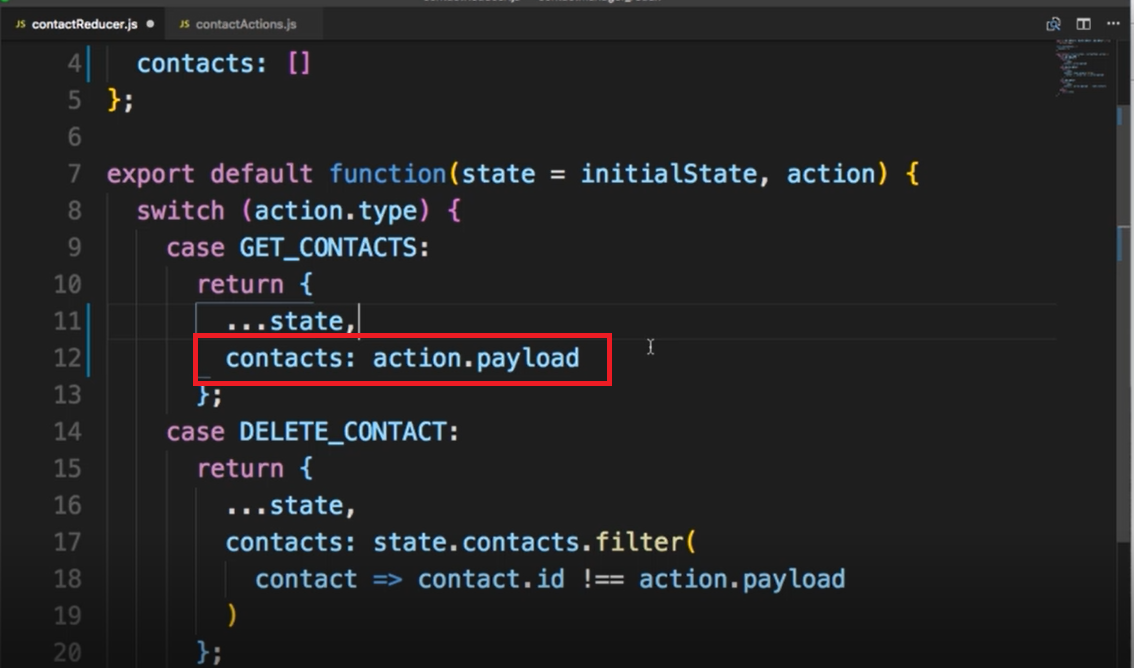




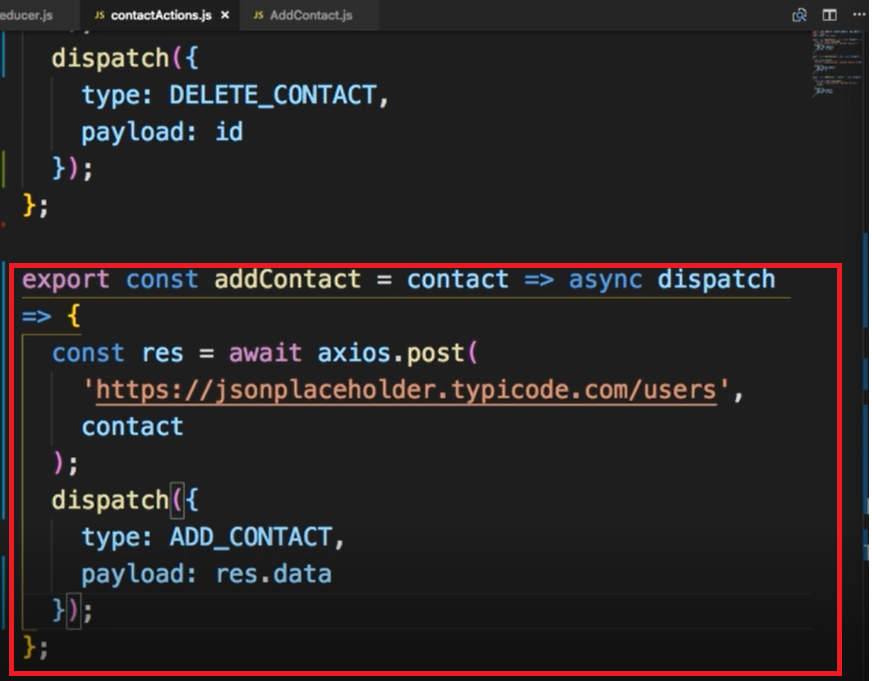
# Calling APIS

## Get API [All the API needs to be called in customAction.js file {contactActions.js}]



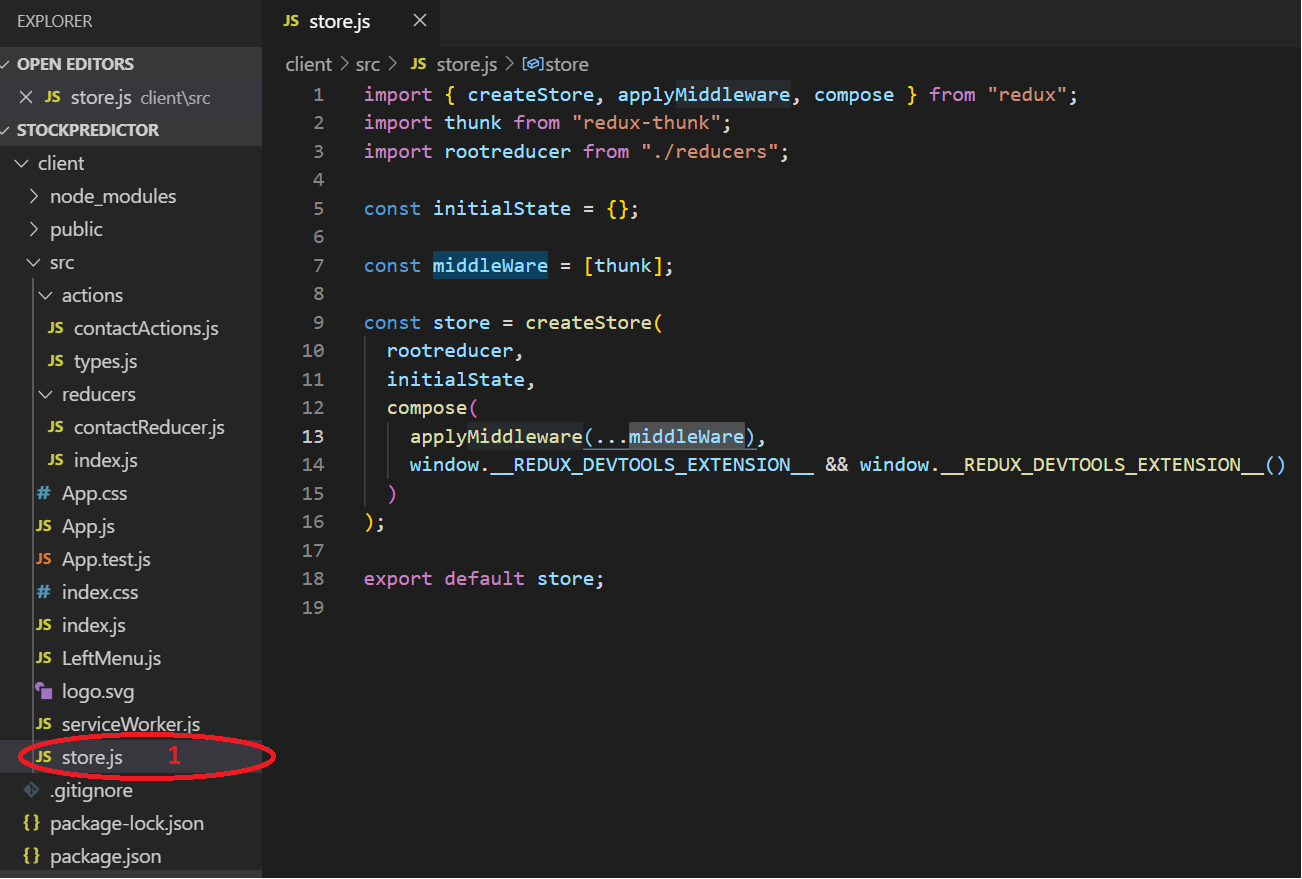


## Post:

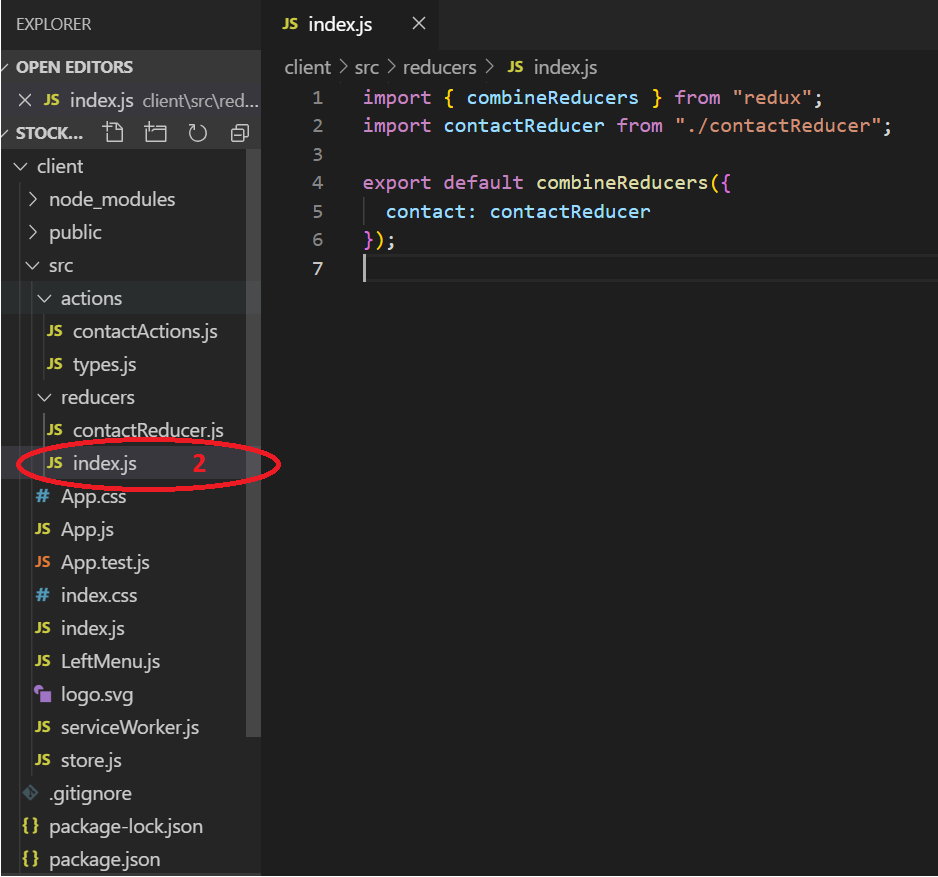


# The Sequence:

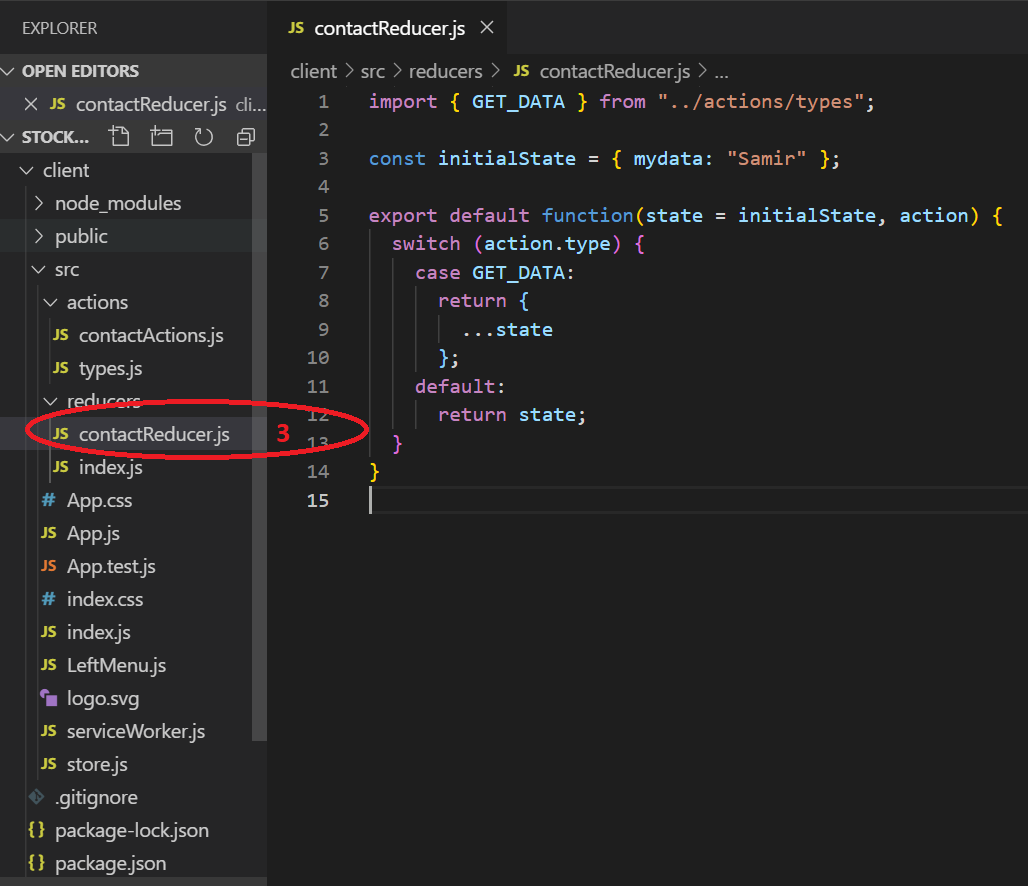
## One



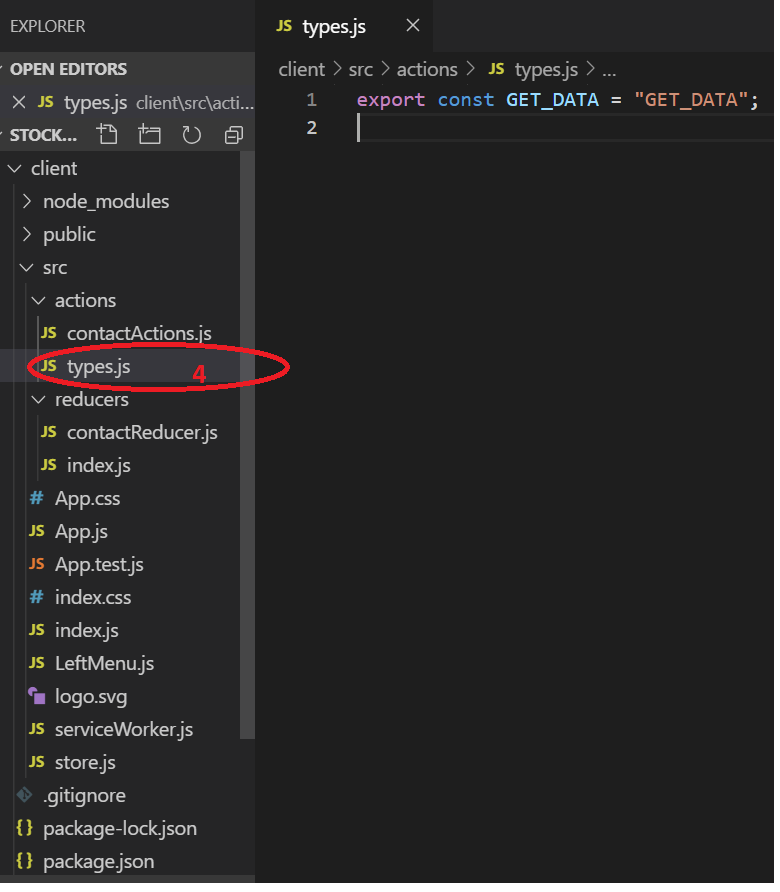
## Two:



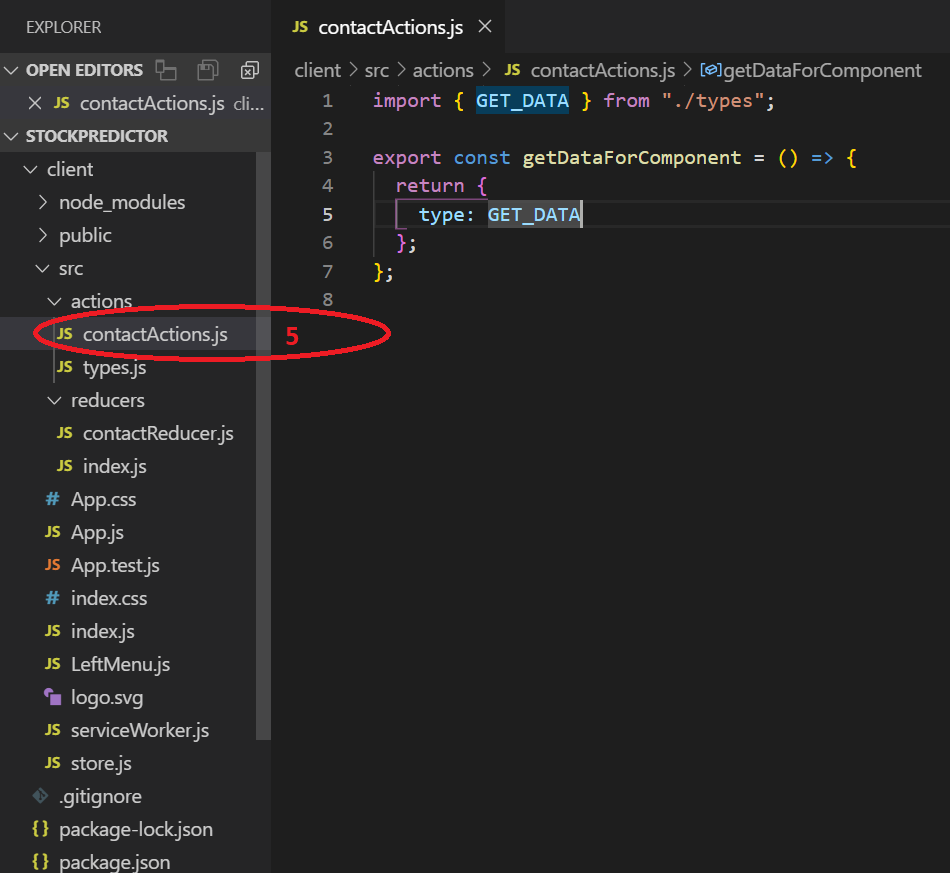
## Three:



## Four:



## Five



## Final

