目录

[1 Demo data passing between components 2](#_Toc30941)

[2 add auth 1: client side - LocalStorage 3](#_Toc24346)

[Step 1: Add localhost to keep track of user login 4](#_Toc26385)

[Step2: create a context to store user status globally 5](#_Toc16276)

[Step3: create ath service 5](#_Toc17285)

[Step 4: add user context wrapper to index.js 6](#_Toc13111)

[Step 5: implement context in app.js 7](#_Toc17851)

[Problem? Didn’t redirect after logged in 7](#_Toc11473)

[3 add logout 8](#_Toc4443)

[4 Convert a class component to functional component 9](#_Toc30028)

[5 Add web3 integration 9](#_Toc14333)

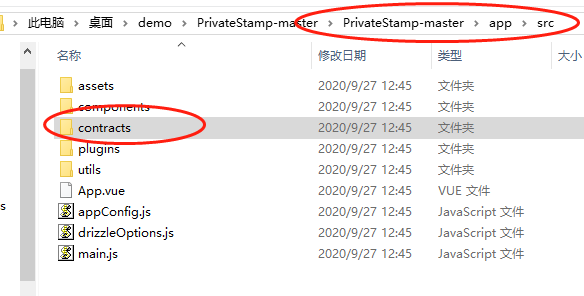
[5.0 explain the code 9](#_Toc31441)

[5.1 use truffle to deploy the contract: 9](#_Toc20045)

[5.2 open remix to interact with the contract 9](#_Toc2526)

[5.4 how to writ test cases 11](#_Toc22354)

[5.3 copy abi to the project folder 11](#_Toc18416)

[ 12](#_Toc7898)

[5.3 2 ways to interact with blockchain 12](#_Toc17598)

[5.4 query data 14](#_Toc9307)

[5.5 add web3 to the frontend 14](#_Toc24090)

[6 Write a Store Factory Using Openzeppelin 15](#_Toc458)

[1coding 15](#_Toc21480)

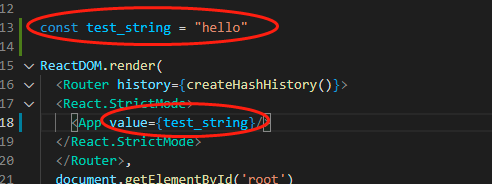
[2 Manual test 16](#_Toc22066)

[3 truffle test 16](#_Toc5332)

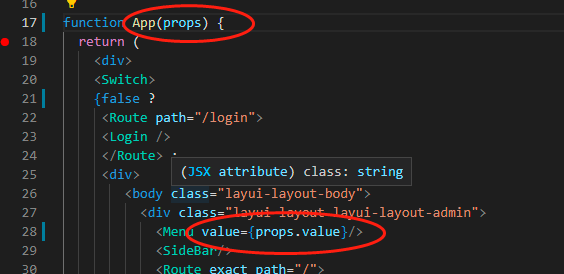
# 1 Demo data passing between components

Goal: pass the value from index.js to menu.js

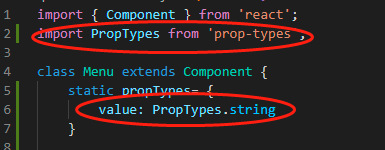
1. Index.js



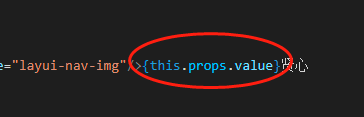
1. App.js



1. Menu.js



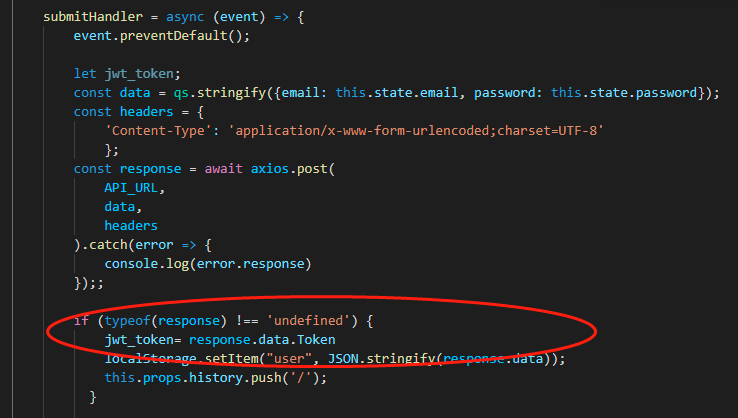
Add variable to the front end



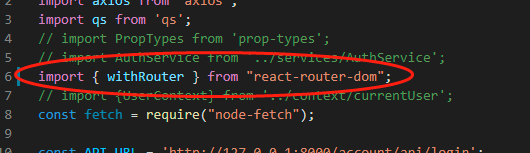
# 2 add auth 1: client side - LocalStorage

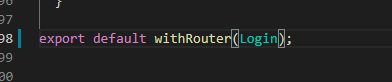
https://github.com/onebit256/batch2-lab7-1-react-auth-client

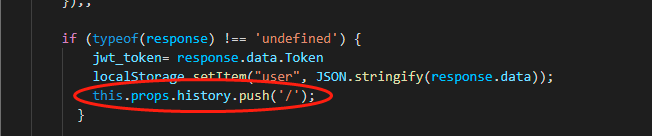
## Step 1: Add localhost to keep track of user login



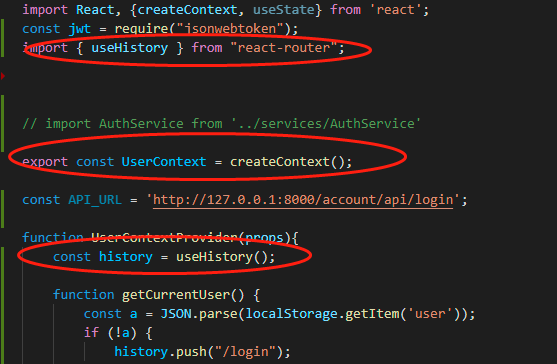
Add route link in code to redirect to the main page after logging in





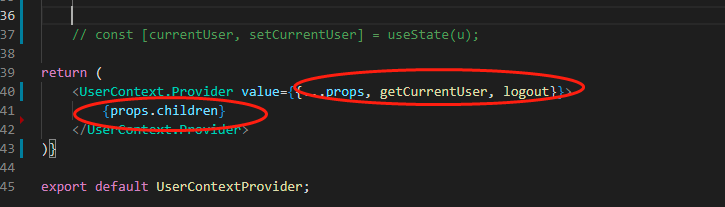


## Step2: create a context to store user status globally



- create context and export it

- import Route History context



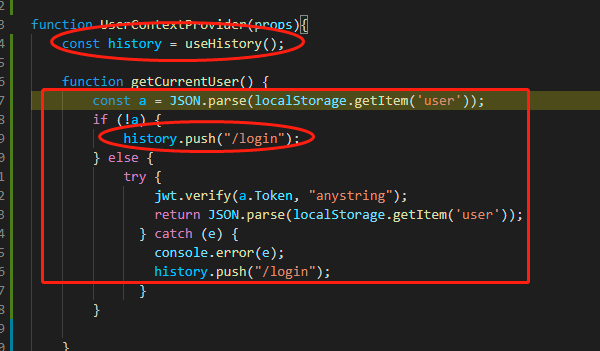
- {...props, currentUser, setCurrentUser} means merge all key value pair with currentUser and setCurrentUser

- props.children is a placeholder for all the content wrapped by this context

## Step3: create ath service

- Add getCurrentUser function to currentUser Context to obtain the user login status in localStorage

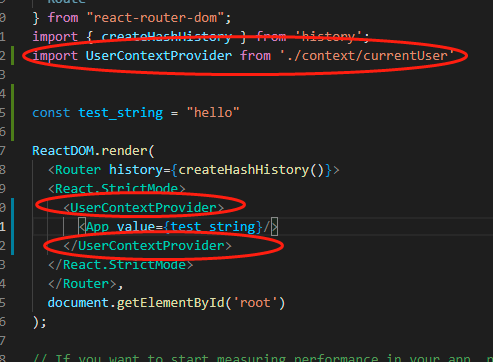
- Add verify everytime we getCurrentUser



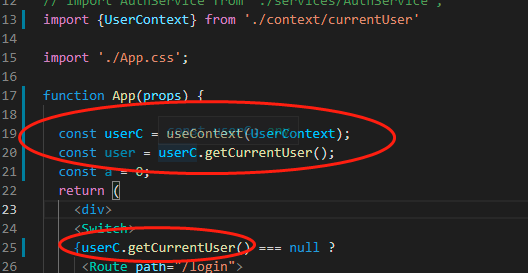
LocalStorage.removeItem: remove the data ; getItem: query data

History.push redirect the path

## Step 4: add user context wrapper to index.js

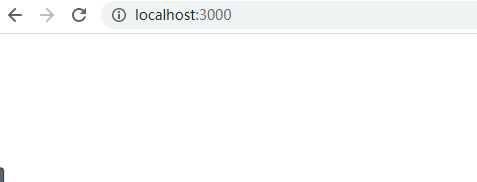


## Step 5: implement context in app.js

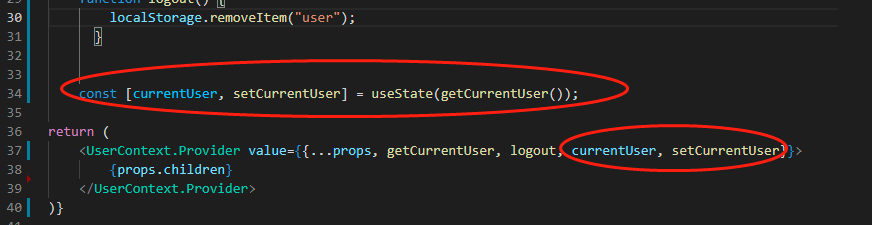


Use useContext hook to invoke getCurrentUSer from context

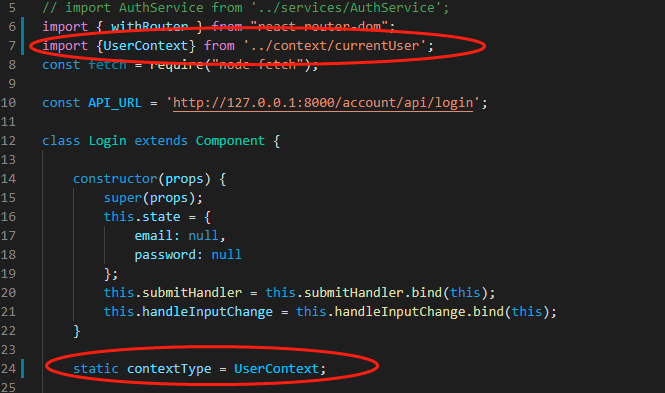
## Problem? Didn’t redirect after logged in

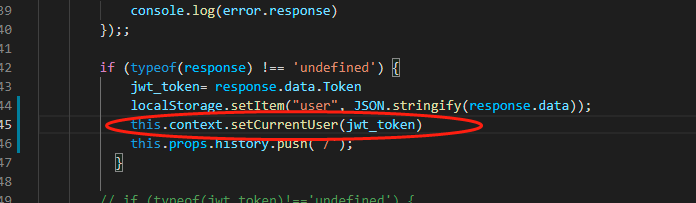


The cause is the webpage didn’t get rendered, so we still need currentUser state and setState to update the page



Use setCurrentUser to update the page, after logged in

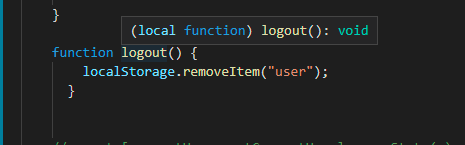




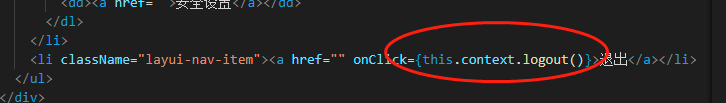
\*Debug and check the data flow

# 3 add logout

Step1: add logout to currentUser Context



Step 2:Use onClick to triger logout function in currentUser context



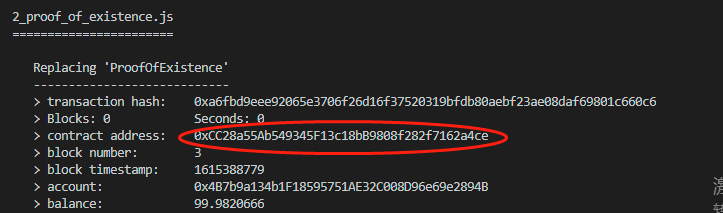
\*Debug and check the data flow

# 4 Convert a class component to functional component

# 5 Add web3 integration

## 5.0 explain the code

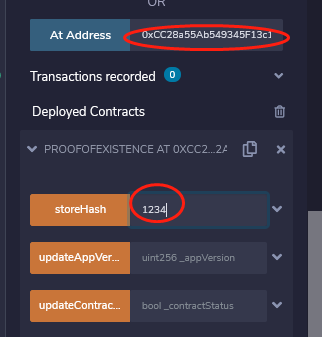
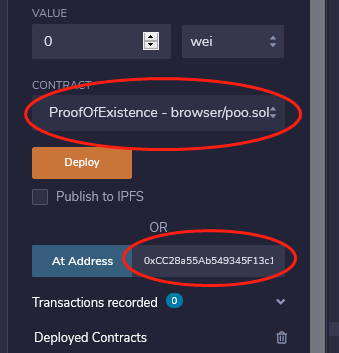
## 5.1 use truffle to deploy the contract:

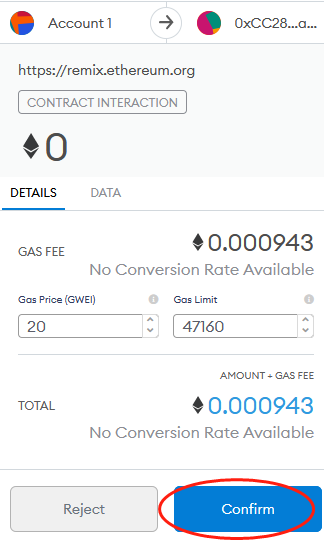


## 5.2 open remix to interact with the contract

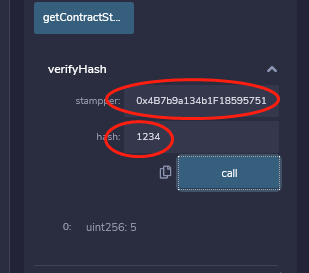
Compile the contract on remix, and use at to deploy the contract

Store hash



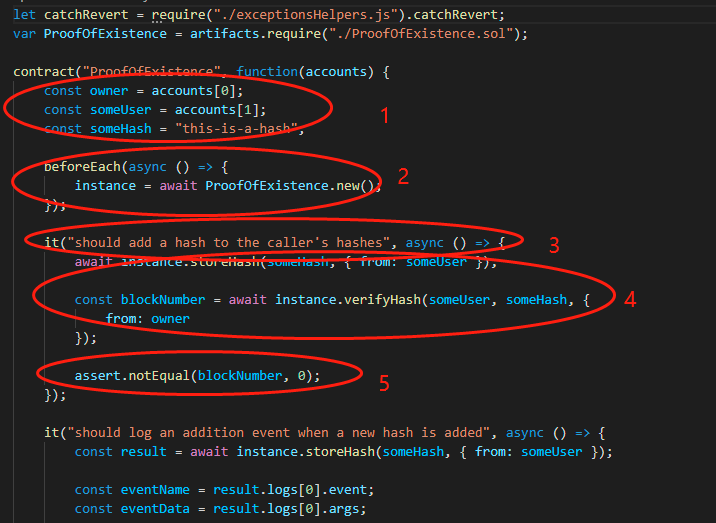


Verify hash



So this transaction gets verified and stores at block 5

## 5.4 how to writ test cases



1set up the accounts

2set up the global environment for all the test cases

3name a test case

4execute the logic

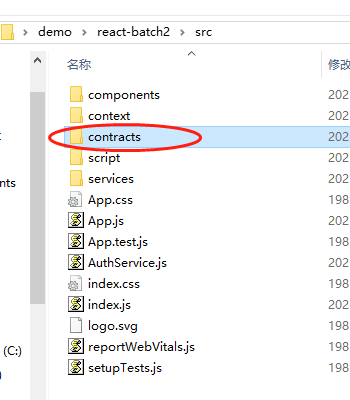
5compare the resultq

## 5.3 copy abi to the project folder

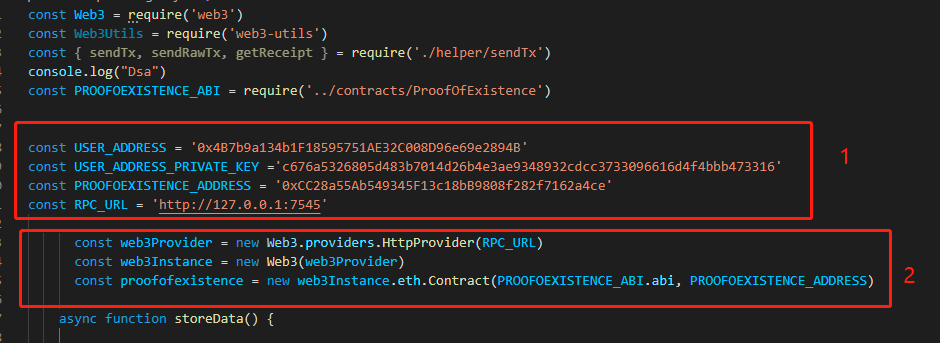
copy

## 

To



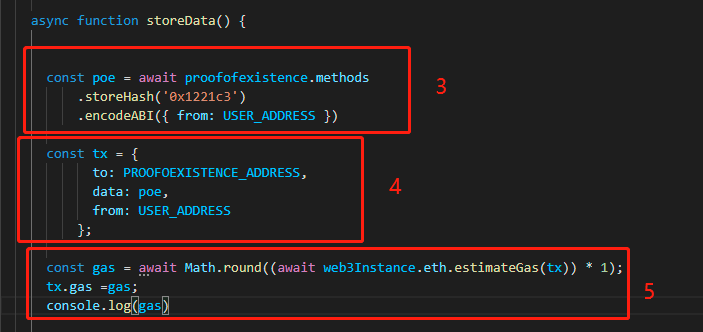
## 5.3 2 ways to interact with blockchain



Step1 : set addresses

Step2: create web3 connector

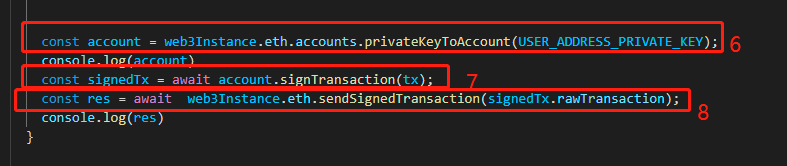
1 web3 send transaction



Step3:get method encode

Step4:assign encode and other params to tx

Step5:get estimate gas

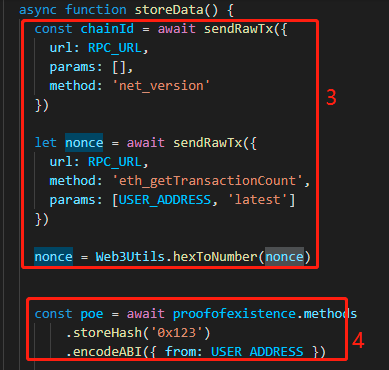


Step6:import web3 eth account from private key

Step7:sign the raw data with pk

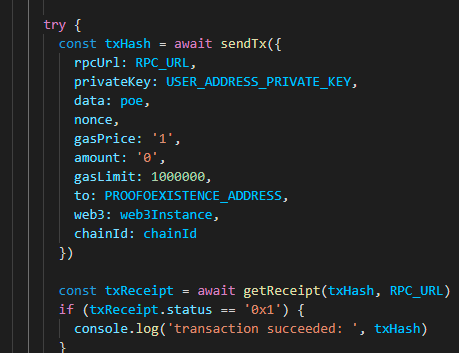
Step8: send signed transaction to blockchain

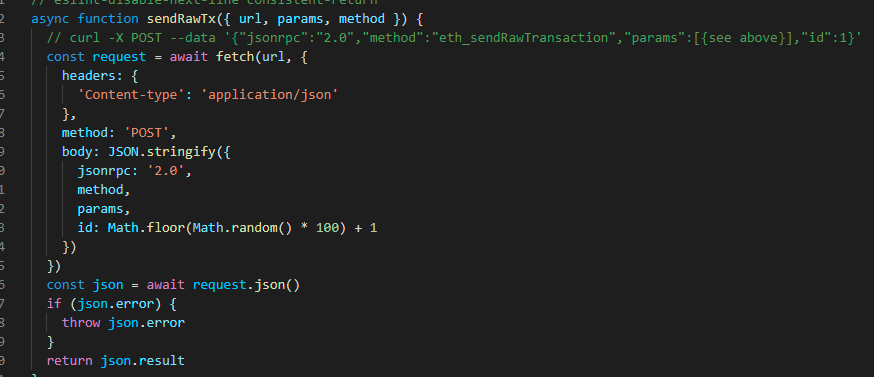
2 RPC post request



Step3: get chainid and nonce

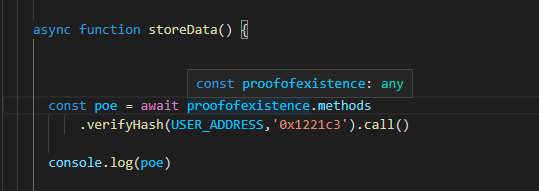
Step4:get methods encode





Use rpc post api to send signed raw data

## 5.4 query data



Use .call()

## 5.5 add web3 to the frontend

# 6 Write a Store Factory Using Openzeppelin

Github.com/onebit256/proxy

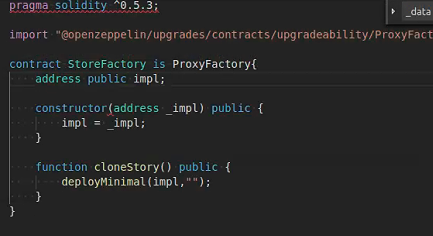
## 1coding

Ganache-cli -m ‘’

Truffle init

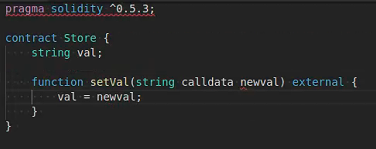
Npm init

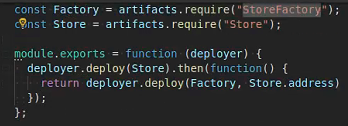
Npm i @openzeppelin/upgrades@2.8.0





As there is no init function





## 2 Manual test

Const sf = await StoreFactory.deployed()

Sf.impl()

Networks

Sf.clonestore()

Sf.getPastEvents()

Sf.clonestore()

Sf.clonestore()

St1 = Store.at(‘’)

St1.setVal(“ss”)

St1.val()

## 3 truffle test