React will re-render when its completely different object.

**State**: only this component can be modify.

**Event Handler:** Every time, this event has been triggered, there is functionality need to happened.

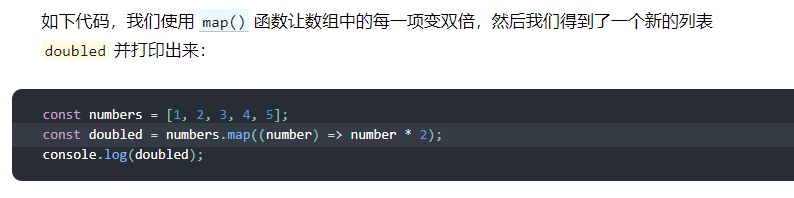
**callBack Function:** pass handler, it want to call some process. (This button want to be called whenever it gotta be changed).

React Render when **state has COMPLETE different object in memory**, then it will re-render.

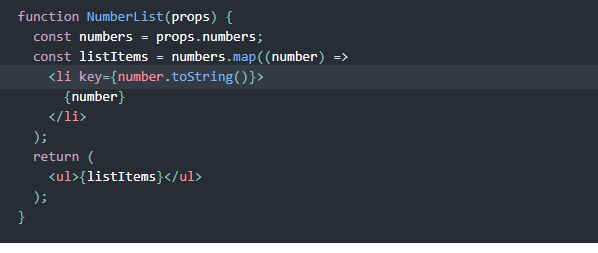
**setState->** doing the shallowmerge when pass a object(find the keys in the state, compare and update the value when they have same keys.)

* **setState(state,props)** -> **state** means the **current state**(update the state based on previous state)

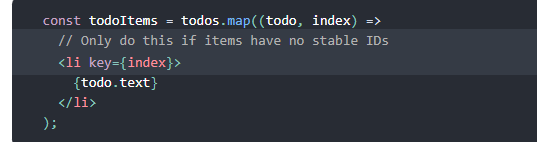
**map**() function:



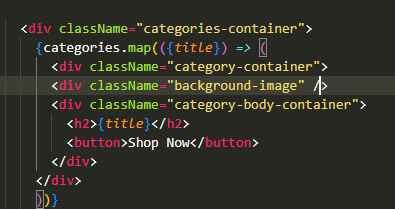
在React中， 使用map function则必须使用 key value(用于确定哪些值被改变



如果没有id，则可以用value的index作为key。



**Map也可以传输component：**

****

**根据特定object，遍历整个array输出该component**

**Single Page Application(SPA)**:

* Send Request to Server
* Server send back HTML,CSS,JS(contain React Code)

**SASS**;

* SCSS 为sass文件
* SASS -> 允许直接使用子类如果该子类在parent class
* backgroundImage = `url(${}`)

**比如：在该子类里面可以直接操控子类下的tag element 渲染**

****

**React-Router:**

* **npm install react-router-dom**
* Import {BrowserRouter} from’react-router-dom’ in index.js
* <BrowserRouter> <App> </BrowserRouter>
* **In App component:** 
  + **Import Router, route**
  + **<routes>**

**<Route path=’/extendName’ element={componentName}>**

**</routes>**

* <outlet /> -> This allows nested UI to show up when **child routes** are rendered.
* **显示两个Component当这两个component使用同一个path (Outlet)**
* <*Route* path='/' element={<*Navigation* />} >
* <*Route* index element={<*Home* />} />
* </*Route*>
* Fragment-> return nothing, do not need rendering all other things. Using Fragment instead of using div
* <Link className to=’/pathName’></Link> (仅限内部链接)
* **Import ReactComponent as logoName from ‘logoPath’;**
  + **<logoName />**
* 获取input值 const Name = this.ref.refName.value
  + **<input** ref=’refName’ type=’text’>
  + **This.ArrayName.push/unshift(Name) -> 来加入新输入的值.**

**Firebase Auth:**

**Google Auth Set Up**

import {initializeApp} from 'firebase/app';

import {getAuth,signInWithRedirect,signInWithPopup,GoogleAuthProvider} from 'firebase/auth';

//configuation

*const* firebaseConfig = {

    apiKey: "AIzaSyBRKd7\_xyg3wcZ75Wy-EjOPH6r9VItrPPQ",

    authDomain: "ecommerce-react-10b16.firebaseapp.com",

    projectId: "ecommerce-react-10b16",

    storageBucket: "ecommerce-react-10b16.appspot.com",

    messagingSenderId: "520440268620",

    appId: "1:520440268620:web:06c5316e6db806adb571e9"

  };

  // Initialize Firebase

*const* firebase = initializeApp(firebaseConfig);

//Require by google

*const* provider = new GoogleAuthProvider();

  //Has to choose an account

  provider.setCustomParameters({

    prompt:"select\_account"

  });

  //Authentication Method, Same for every auth method

  export *const* auth = getAuth();

  //Create a method that it will sign in with google, getting the authentication method with google provider

  export *const* signInWithGooglePopup = () *=>* {

    return signInWithPopup(auth,provider);

  }

在其他componet使用google Auth

*const* logGoogleUser = async () *=>* { //Use method from other file, choose async

*const* response = await signInWithGooglePopup();

        console.log(response);

    }

Write in data into firestore database: (firebase JS)

export *const* db = getFirestore(); //direct point to our database

  export *const* createUserDocumentFromAuth = async (*userAuth*) *=>* {

*const* userDocRef = doc(db,'users', *userAuth*.uid);   //doc took 3 indexes, database, collection, uid

    console.log(userDocRef);

*const* userSnapshot = await getDoc(userDocRef); //get data related to the document, speific object.

    console.log(userSnapshot.exists()); // Verify that this data exist or not.

    //if user data exists, return userDocRef.

    //If not, we can create 1.

    if(!userSnapshot.exists()){ // If not exists

*const* {displayName,email} = *userAuth*;

*const* createdDate = new *Date*(); //when signed In

        try{

            await setDoc(userDocRef,{ //Create 1 userDocRef with name, email and date.

                displayName,email,createdDate

            })

        }catch(error){

            console.log(error.message);

        }

    }

    return userDocRef;

In Other component

*const* logGoogleUser = async () *=>* { //Use method from other file, choose async

*const* {user} = await signInWithGooglePopup();

*const* userDocRef = await createUserDocumentFromAuth(user);

    }

读取input值，并更新其state：

 <form onSubmit={() *=>* { }}>

                <label >Name</label>

                <input  type="text" onChange={handleChange} placeholder="Name" required name='name' value={name}/>

由于有多个input值，把他们都存入一个object，

使用usestate，把初始值设置该obj，并且把input拆分为一个新的obj、

使用onChange来读取input值比如，event.target为输入的input值

SetState更新每个input中name attribute的值为value

*const* defaultFormFields = {

    name:'',

    email:'',

    password:'',

    confrimpassword:''

}

*const* Register = () *=>* {

*const* [formFields,setformFields] = useState(defaultFormFields);

*const*{name,email,password,confrimpassword} = formFields;

    console.log(formFields);

*const* handleChange=(*event*) *=>* {

*const* {name,value} = *event*.target; //target give the input info

        setformFields({

            ...formFields,

            [name]:value

Reducers : object and action(type:String, payload)