QUẢN LÝ USER

Xử lý component AddUser form:

+ Tạo state trangThaiChinhSua (true),

- Chức năng: Click vào “Thêm mới” thì hiện ra form:

+ Tạo arrow function hienThiForm(),nếu trangThaiChinhSua === true thì return lại Form, false thì không làm gì hết.

+ Xuống hàm render() gọi hàm hienThiForm().

- Chức năng: Click nút “Thêm mới” thì nó hiện nút “Đóng”, click nút “Đóng” thì nó hiện “Thêm mới”:

+ Tạo arrow function hienThiNut(), nếu trangThaiChinhSua === true thì return nút “Đóng lại” và ngược lại.

+ Tạo function thayDoiTrangThaiChinhSua(), để thay đổi state trangThaiChinhSua thông qua hàm setstate(), thangThaiChinhSua : !this.state.trangThaiChinhSua.

+ Click vào nút “Đóng” hoặc “Thêm mới” thì gọi hàm arrow thayDoiTrangThaiChinhSua().

TẠO DỮ LIỆU JSON và IMPORT VÀO COMPONENT <App/> :

* Tạo file .json

+ Mảng người dùng: id, name, permission.

* Đẩy dữ liệu vào <App>:

+ import DataUser from ‘./Data.json’

* Tạo state ‘data’ và gán DataUser cho state ‘data’ vừa tạo thông qua hàm setstate().

IN DỮ LIỆU RA <ListItems>:

* Truyền state ‘data’ từ <App> vào <ListItems> thông qua props ‘data’. (props ‘data’ là mảng các phần tử user sẽ được in ra trong <ListItems/>) (Muốn in ra các phần tử cần tìm kiếm thì truyền các phần tử ấy vào cho props ‘data’).
* Trong component <ListItems> , viết arrow function mappingDataUser(), gọi hàm ở dưới.
* Trong arrow function mappingDataUser(), sử dụng hàm map để truyền value vào trong các <Item>:
  + mappingDataUser = () => this.props.data.map((value, key) => (

<Item userName ={value.name} stt = {key} permission ={value.permission} > </Item>

))

* Trong component <Item>, viết arrow function permissionShow(), if (this.props.permission === 2 ) return “High”;
* Gọi function permissionShow() trong thẻ <tr> permission.

CHỨC NĂNG TÌM KIẾM (kết nối component cha với component con thông qua props):

Lưu ý: tất cả các hàm xử lý logic dữ liệu đều viết ở component bố <App>.

* Lấy được từ khóa tìm kiếm trong component <Search> rồi đẩy về component <App>, sau đó so sánh dữ liệu rồi truyền vào <ListItems> rồi in ra màn hình:

+ Bước 1: cần phải kết nối component <App> với <Search> thông qua hàm được viết trong component <App> và truyền thông qua props getTextSearch(dl).

+ Bước 2: lấy text trong ô Search về:

* Khai báo arrow function isChange(event).
* Sử dụng: event.target.value = text điền vào.
* Gọi hàm ra trong sự kiện onChange.
* Tạo state tempValue để lưu trữ dữ liệu tìm kiếm.
* Điền text hoặc khi nhấn Submit thì lấy được text:
  + Viết arrow function isChange để lấy text:
    - isChange = (event) => {

console.log(event.target.value)

* + - * Dùng hàm setState để gán event.target.value cho state tempValue.

(Lấy được text tìm kiếm, lưu vào state ở component <Search>)

* + - * this.props.getTextSearch(this.state.tempValue);

}

* + Gọi Arrow function isChange (event) qua xử lý sự kiện onChange ở thẻ <input>.
  + Muốn mỗi lần gõ phím thì cập nhật các phần tử thỏa mãn thì trong hàm isChange(),gọi hàm props this.props.getTextSearch(this.state.tempValue).

+ Bước 3: Truyền dữ liệu ấy về <App/> từ component <Search>:

* Có text rồi truyền về <App>:
  + Lưu ý: Bố muốn truyền dữ liệu cho Con thì thông qua Props, Con muốn truyền dữ liệu cho bố thì thông qua THAM SỐ.
  + Viết 1 arrow function getTextSearch(dl) ở <App> sau đó truyền cho <Search> thông qua props và lưu giá trị ấy ở trong state để so sánh từ khóa để lấy ra dữ liệu cần tìm:
    - Tạo state ‘searchText’.
    - getTextSearch = (dl) => {
      * Dùng hàm setState() để lưu giá trị tham số dl vào state ‘searchText’.

}

* + - Viết trên trong hàm Render(), trên hàm Return(),so sánh tên cần tìm với từng phần tử trong mảng dữ liệu , dùng hàm forEach() :
      * + Lưu ý: Khi tìm kiếm gì đó thì có thể có rất nhiều phần tử thỏa mãn, nên không thể tạo 1 state trung gian để lưu được mà phải tạo 1 mảng các phần tử để lưu lại hết các phần tử thỏa mãn.
        + Var ketQua = [];
        + Trong <App>:

this.state.data.foreach((item) => {

If(item.name.indexOf(this.state.searchText) !== -1){

ketQua.push(item);

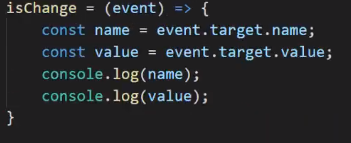
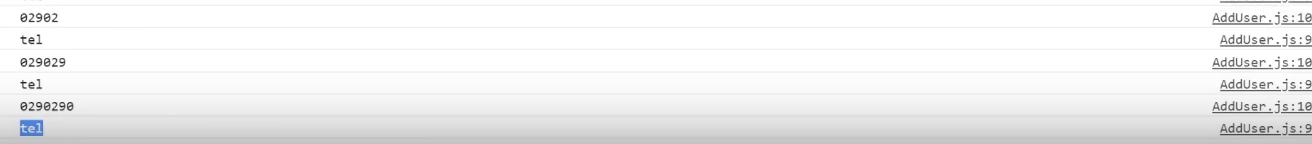
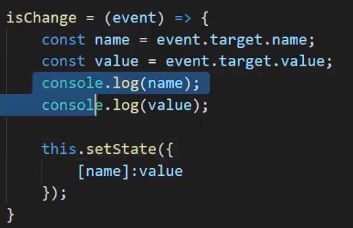
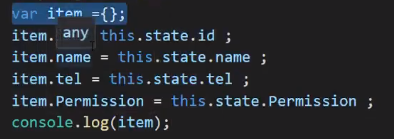
}

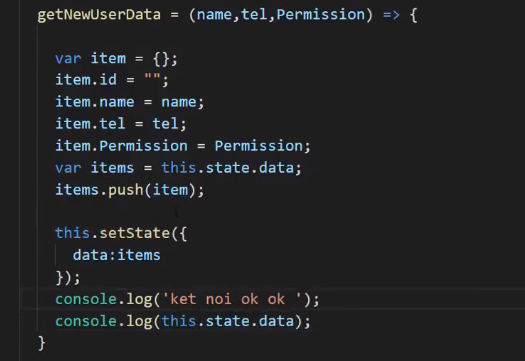
})

* + ở <Search/>: <div onClick = {(dl) => this.props.getTextSearch(this.state.tempValue)}></div>

CHỨC NĂNG THÊM MỚI THÀNH VIÊN (<AddUser/>):

Gồm 3 bước:

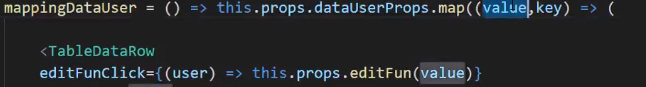
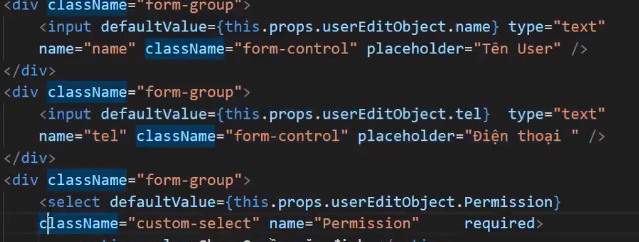
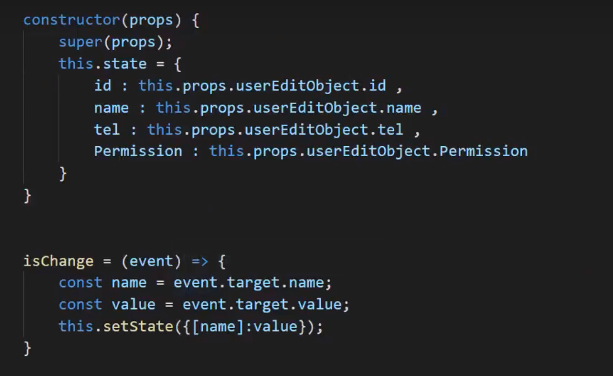
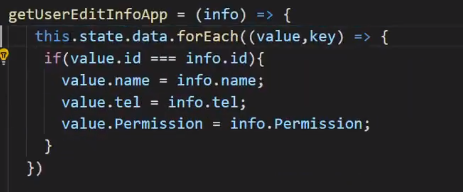
* Bước 1: Điền nội dung, ấn nút ‘Thêm mới’ thì lấy được nội dung
  + Sử dụng sự kiện onChange để lấy dữ liệu (even.target.value, event.target.name):
    - Viết hàm isChange(event):
    - 
    - 
    - Event.target.name chính là Attribute ‘name’ trong thẻ <Input> do mình đặt.
  + Đặt dữ liệu trả về vào state của chính nó ( mục đích là lưu trữ ở Component đó và sử dụng khi cần):
    - Tạo các state ‘id, name,permission’.
    - Dùng hàm setState để set cho mảng [name] bằng value:
      * 
      * Mảng [name] ở đây là giá trị chuỗi mình đạt trong attribute name:
        + C:\Users\User\Desktop\a.png
    - Sau đó, tạo đối tượng item rỗng.
    - Gán các State id, name, permission cho đối tượng Item vừa tạo.
      * 
* Bước 2: Lấy được nội dung tel, name, id , gửi về <App/>, sau đó đóng gói thành đối tượng:
  + Kết nối <App> với <AddUser> thông qua props:
    - Tạo arrow function getNewUserData(name,tel,permission) ở <App> để nhận các thông tin của users như name, tel, permission.
    - Truyền arrow function getNewUserData(name,tel,permission) xuống <AddUser> thông qua props getNewUserData.
    - Sau khi nhận được các thông tin ở <App>, đóng gói thành 1 đối tượng {}User.
* Bước 3: <App> nhận được đối tượng User, đẩy vào dữ liệu json, sau đó in ra giao diện dữ liệu mới.
  + Sau khi nhận được đối tượng item rồi, tạo biến ‘items’ = this.state.data.
  + Dùng hàm Push() để đẩy đối tượng ‘item’ vào ‘items’.
  + Dùng hàm setState() để cập nhật state ‘data’ bằng ‘items’.
  + Sau đó nó sẽ tự in ra giao diện dữ liệu mới thông qua props ‘data’ và hàm MappingUserData().
  + Muốn xóa trắng nội dung sau khi thêm mới:
    - Các form đó phải là thẻ <input>.
    - Có Type = “reset” , value =”Thêm mới” nằm trong nút “Thêm mới” or “đăng ký” (là cái nút submit á).
    - Nằm trong vùng 1 cái Form.



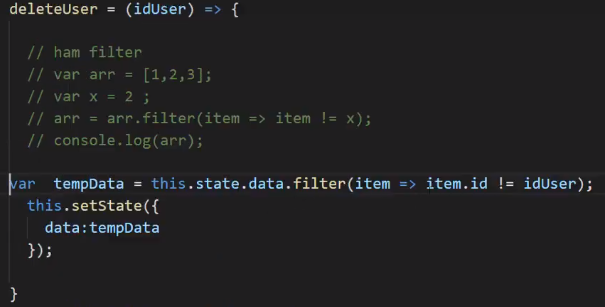
* + Tự tạo id không trùng lắp thông qua UUID:
    - <https://www.npmjs.com/package/uuid>
    - npm install uuid
    - khởi tạo biến const uuidv1 **=** require('uuid/v1') ở file sử dụng.
    - gọi uuidv1();
    - 
    - Đây là Id lưu trong state data, số lung tung không theo thứ tự 1 2 3 , còn STT in trên màn hình dựa theo {key, value} của hàm map() nên theo thứ tự 1 2 3.

CHỨC NĂNG SỬA THÔNG TIN (EDIT) THÀNH VIÊN :

Gồm 4 bước: **Arrow function EditUser():**

* Bước 1: kết nối <App> => <TableData/> => <TableDataRow/>, **Bước 2**: Click vào nút ‘Sửa’ , lấy được thông tin User cần sửa:
  + Tạo arrow function EditUser(user), truyền từ <App> => <TableData> => <TableDataRow> thông qua props EditUser.
  + Trong <TableDataRow>, tạo arrow function editClick().
  + Trong editClick(), gọi this.props.EditUser(user).
  + Sử dụng hàm OnClick = {this.editClick()} để xử lý xự kiện Click.
  + Lưu ý về tham số User: tham số User vẫn truyền bình thường, nhưng lưu ý ở hàm mapping thì truyền ‘Value’ chứ không phải là ‘user’.
    - 
* Bước 3: Ấn nứt Sửa,tạo Form và Load nội dung cần sửa vào Form:
  + ở <App>,tạo State ‘editUserStatus’: false.
  + Tạo Arrow Function changeEditStatus():
    - Dùng hàm SetState() để phủ định giá trị của State ‘editUserStatus’.
  + Truyền Arrow Function changeEditStatus() từ <App> => <EditUser>, nút ‘Lưu’, hàm EditClick ở <TableDataRow>.
  + Nói Chung: TỰ LÀM chức năng khi ấn Edit thì hiển thị Form Edit, dựa trên state ‘editUserStatus’ để xác định hiển thị form Edit hay sao, tự làm thông qua truyền các state và props, đã học hết.
  + Khi ấn Edit, hiên thị form và các THÔNG TIN MẶC ĐỊNH BAN ĐẦU của User đó thì ta làm như sau:
    - Tạo state userEditObject.
    - Trong hàm EditUser(user), dùng setState() để lưu trữ tham số ‘user’ vào state userEditObject.
    - Truyền state userEditObject ấy thông qua props vào thẻ <EditUser>.
    - Xác định thuộc tính DefaultValue bằng các props ‘userEditObject’ vừa được truyền trong thẻ <Input>
    - 
* Bước 4: Ấn Submit và lưu thông tin về <App> và hiển thị ra màn hình:
  + Khi ấn nút Lưu, thì lấy thông tin cần lưu:
    - Tạo các state ‘id, name, tel, permission’ để lưu thông tin trong quá trình sửa thông tin(giống như THÊM MỚI THÀNH VIÊN).
    - Tạo Arrow Function isChange(event).
    - Gọi Arrow Function isChange thông qua sự kiện onClick ở các thẻ <Input>
    - 
  + Gửi dữ liệu đã sửa vào <App> (chỉ là gợi ý thui tự suy nghĩ từ đâu):
    - Kết nối <App> với <EditUser> thông qua arrow Function getUserEditInfo(info) bằng cách truyền các props.
  + Trong <App/>, lấy dữ liệu mới đó cập nhật vào state ‘data’:
    - Hiện tại, ở <App>, nhận được ‘info’ rồi thông qua hàm getUserEditInfo(info).
    - Trong getUserEditInfo(info):
      * Dùng hàm forEach(), để so sánh từng phần tử trong state ‘data’ với id của ‘info’,nếu trùng thì cập nhật lại state ‘data’.
      * 

CHỨC NĂNG XÓA USER:

* Bước 1: Ấn nút xóa, lấy được id của user, truyền id lên <App>::
  + Kết nối <App> và <TableDataRow> thông qua arrow Function deleteUser(idUser).
* Bước 2: App sẽ tìm trong state ‘data’ và xóa:
  + Giống chức năng trên, dùng hàm Filter() để so sánh id của từng phần tử trong data với idUser, khác thì nó lọc, xóa ra khỏi mảng.
* 

3 HÀM THAO TÁC VỚI LOCALSTOR:

https://www.w3schools.com/html/html5\_webstorage.aspAGE

* Cấu trúc:
  + Key.
  + Value.
* Hàm thao tác, gồm 3 hàm:
  + Tạo:
    - setItem.
    - localStorage.setItem(“Key1”, “haha “).
  + Xóa:
    - removeItem.
    - localStorage.removeItem(“Key1”).
  + Lấy Thông tin:
    - getItem.
    - localStorage.getItem(“Key1”).
  + **Lưu ý:** trong value của LocalStorage chỉ lưu các STRING, trong khi data của mình là các ĐỐI TƯỢNG , nên ta cần phải dùng hàm JSON.stringify(DataUser) để chuyển các ĐỐI TƯỢNG ấy thành các STRING(chuỗi), truyền vào storage thì đối tượng => chuỗi(STRINGIFY), lấy ra từ storage thì chuỗi => đối tượng(PARSE).
    - JSON.parse(mảng đối tượng): chuyển các CHUỖI thành các ĐỐI TƯỢNG.
    - JSON.stringify(mảng đối tượng): dùng để chuyển các ĐỐI TƯỢNG User trong file JSON thành CHUỖI.
* Ứng dụng 3 hàm vào project Quản lý User:
  + Tự làm hoặc coi lại. (89, 90)