|  |  |  |
| --- | --- | --- |
| Macintosh HD:Users:hoangnguyen:Documents:TDC:fitlog_blue.png | TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC  **Khoa Công Nghệ Thông Tin** | Macintosh HD:Users:hoangnguyen:Documents:TDC:logoTDC_blue.png |
|  |  |  |

**CHUYÊN ĐỀ PHÁT TRIỂN WEB 1**

| HKI – [2020 – 2021]

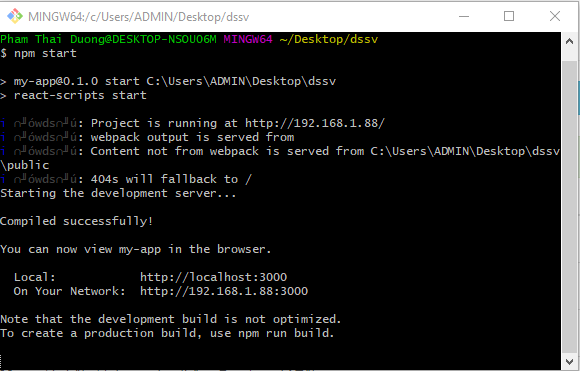
**Hướng dẫn sử dụng Demo React**

**Nhóm I:**

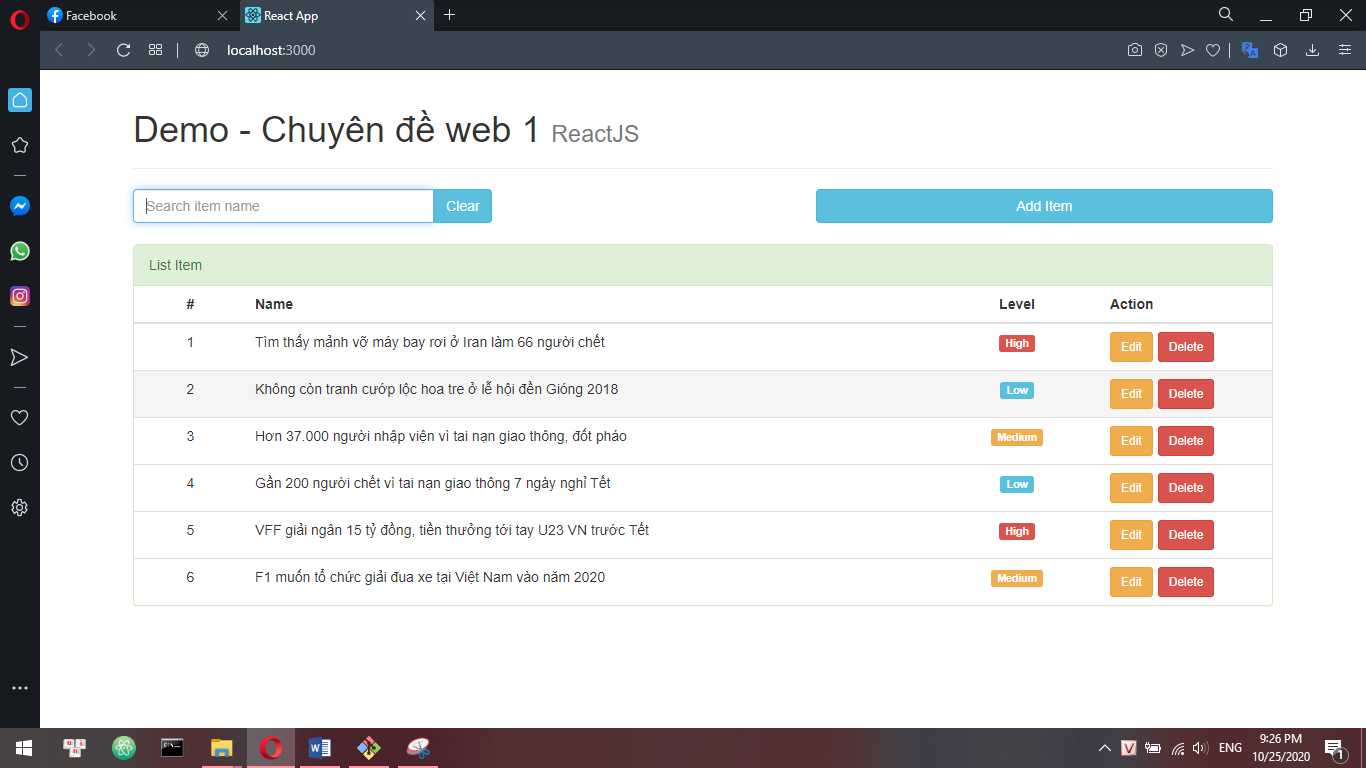
* **Huỳnh Nguyên Vũ**
* **Lê Thành Hưng**
* **Nguyễn Tăng Vĩnh**
* **Phạm Thái Dương**

1. **Cách chạy phần mềm**

* **Phần mềm chạy trên trình duyệt và cần chúng ta cài Node Js và Git (bản mới nhất)**
* **Mở “Git Back Here tại thư mục source code” và gõ “npm start” để chạy chương trình**



* **Giao diện khi chạy chương trình như sau:**

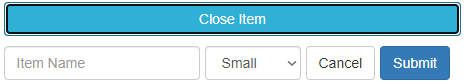


1. **Chi tiết về các chức năng**
2. **Chức năng thêm vào danh sách**

* **Nhấn vào “Add Item” góc bên phải phía trên màn hình để tiến hành thêm**



* **Giao diện thêm bao gồm: Nút đóng , Tên , Mức độ, Hủy thao tác, Submit.**



**Ví Dụ khi thêm tên một người sẽ hiển thị ra Danh sách**



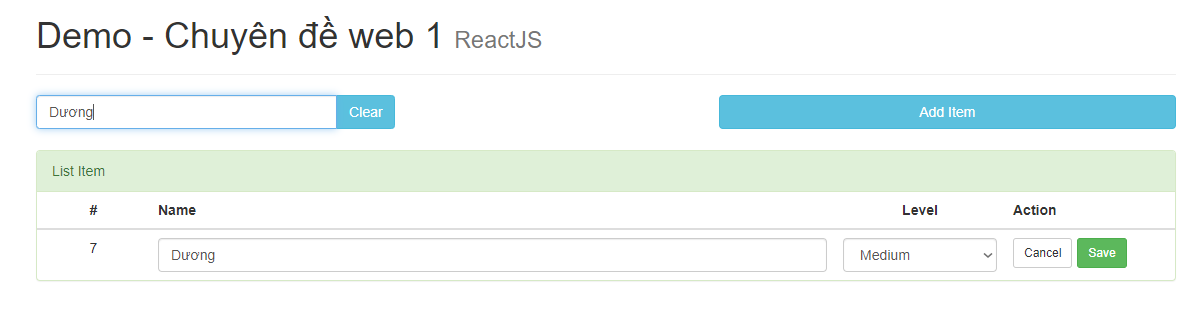
1. **Chức năng chỉnh sửa**

* **Khi nhấn vào Button “Edit”**  **sẽ hiện ra form thông tin item chúng ta chọn.**
* **Có thể chỉnh sửa tên, mức độ**
* **Bao gồm nút Hủy thao tác**  **và nút Lưu sau khi đã chỉnh sửa** 

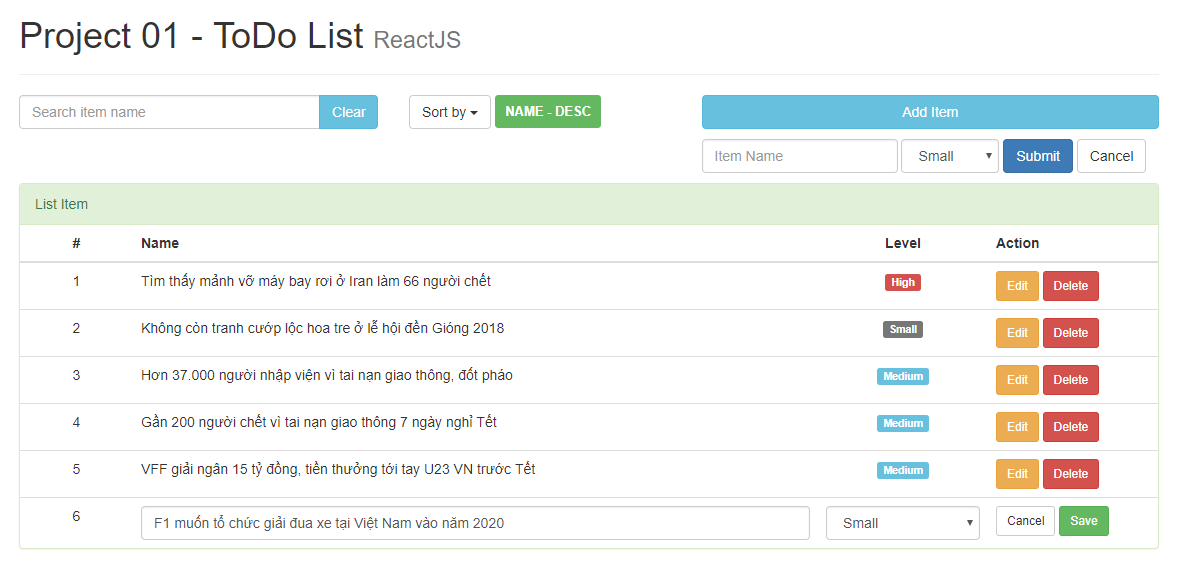
1. **Chức năng tìm kiếm**

* **Khi ta gõ vào ô tìm kiếm** 
* **Sẽ tự động tìm kiếm theo tên những gì có liên quan với tên của danh sách sẽ được tự động hiển thị ra**
* **Button “Clear” dùng để xóa những gì ta vừa mới nhập**

**Ví dụ : Tìm tên “Dương”**



1. **Hướng dẫn code**
2. **Tạo giao diện HTML và CSS**

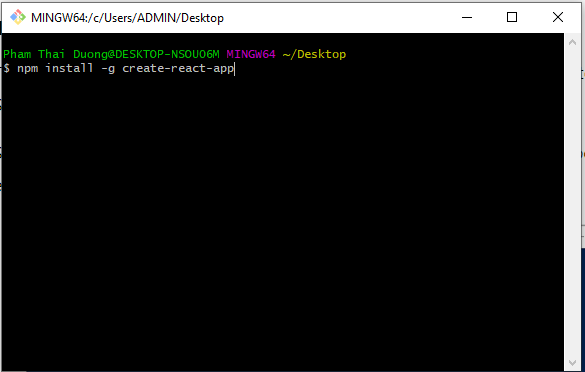


1. **Tạo project và copy giao diện.**

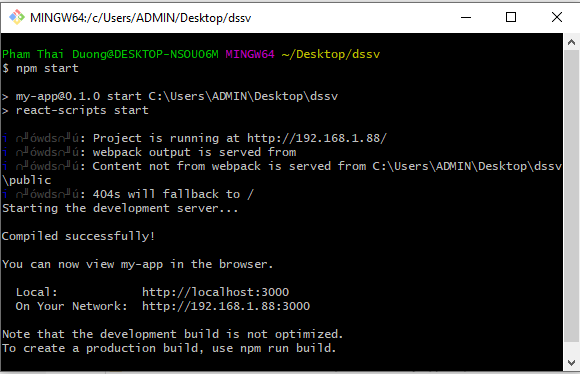
* **Phần tạo project:**

**Tải source giao diện trên driver nhóm i**

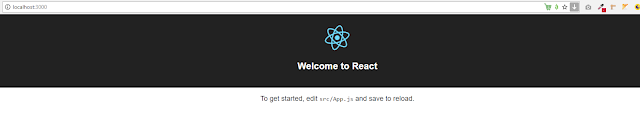
**Gõ lệnh “**npm install -g create-react-app**”**



* **Gõ lệnh “**create-react-app dssv1**” để tạo thư mục , như của mình tạo là dssv 1**
* **Chờ chương trình tải về cho bạn xong thì vào thư mục vừa tạo mở “Git Back Here” và gõ lệnh “**npm start**” để chạy chương trình**

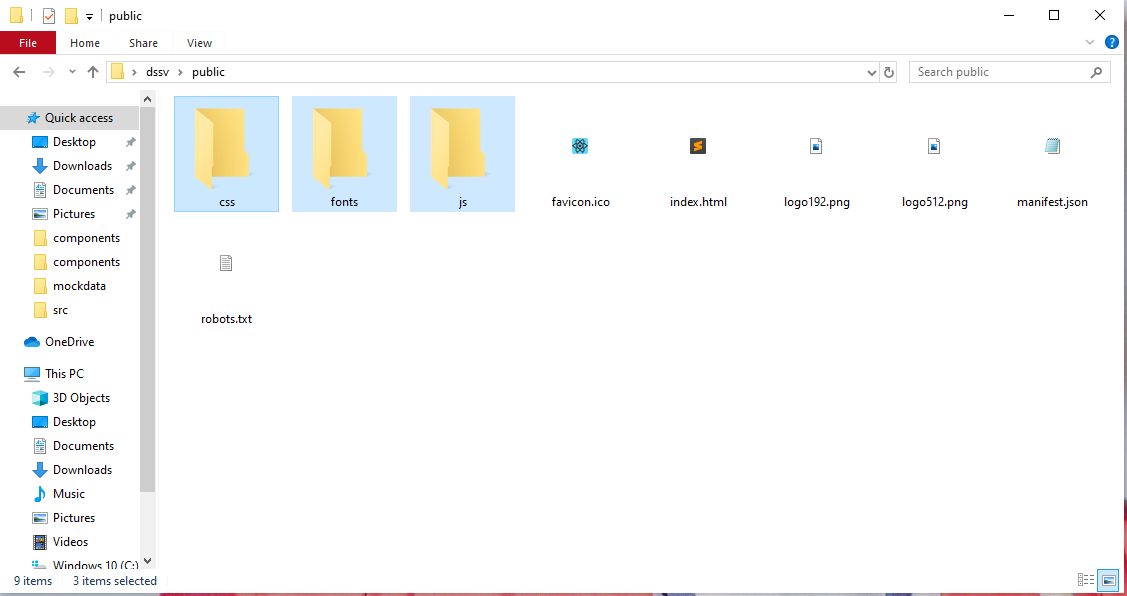


* **Trình duyệt sẽ tự động bật lên một trang web với địa chỉ** [**http://localhost:3000**](http://localhost:3000) **giao diện đầu tiên của trang web**



* **Copy giao diện HTML vào project**

**Chúng ta copy 3 thư mục css, font, js để vào trong thư mục public của file vừa tạo là dssv**

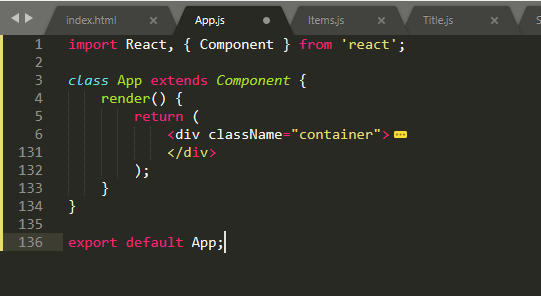


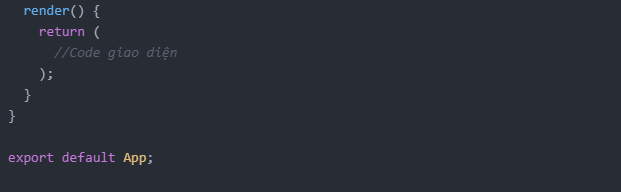
* **Với file index.html của project ta phải thêm các đường dẫn đến file cần thiết sẽ như hình dưới:**



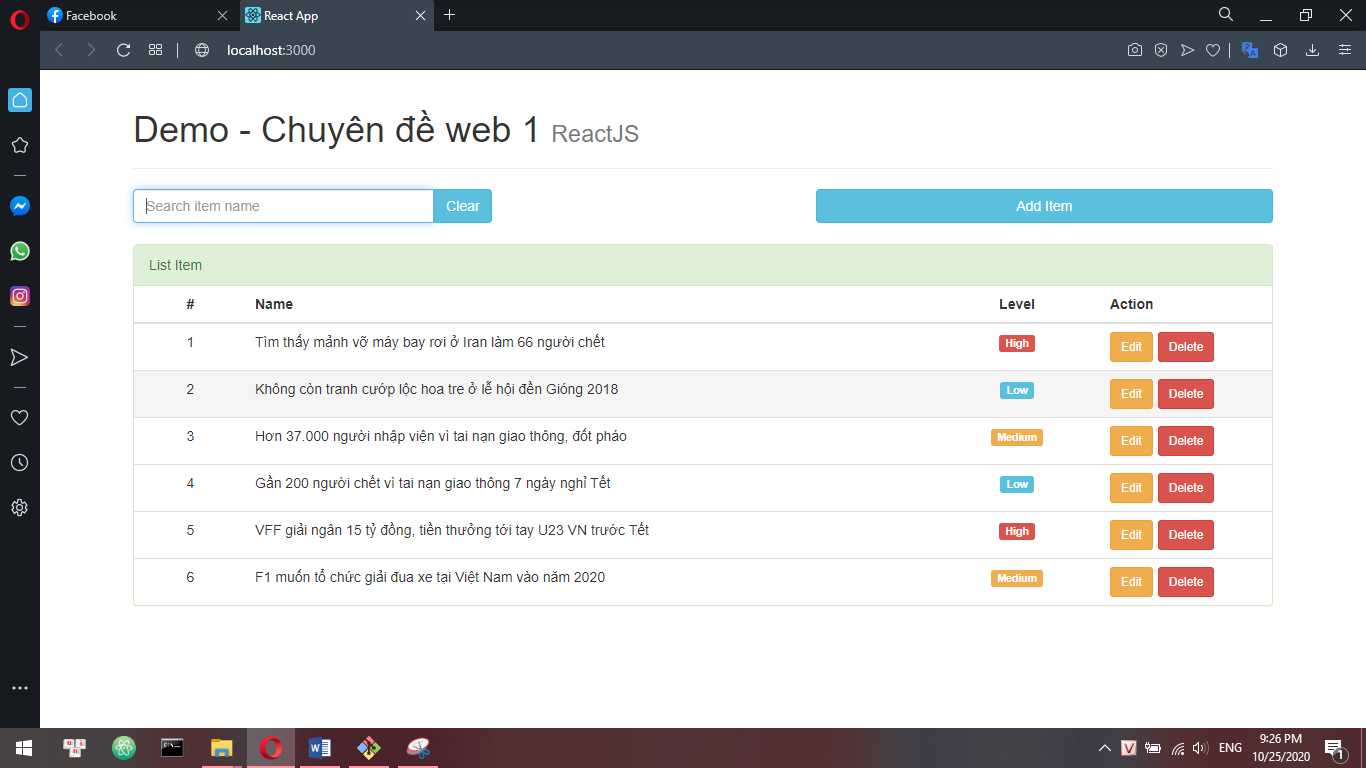
**Bước tiếp theo chuyển toàn bộ code HTML sang JSX:**

* **Vào trang** <https://magic.reactjs.net/htmltojsx.html> để chuyển tự động, vào trang giao diện của hmtl và copy div với class là container để chuyển.
* **Sau khi chuyển JSX thì mở file App.js nằm trong folder src và dán vào giao diện:**





**Nếu làm đúng thì chúng ta sẽ có giao diện trên trình duyệt.**

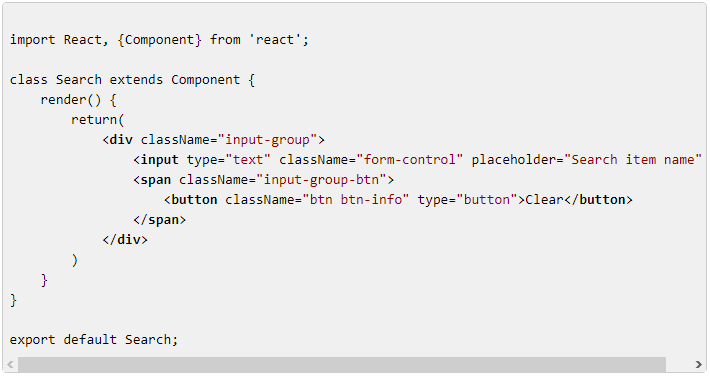


1. **Phân chia component**

* **Mình sẽ tách một lớp component và gọi nó vào trong App.js**
* **Để dễ quản lí ta sẽ tạo thêm một folder bên trong src tên là component**
* **Tạo file có tên Title.js để hiển thị tiêu đề project**



* **Tạo thêm file Search.js để chứa giao diện tìm kiếm**



* **Tạo file có tên Sort.js để chứa giao diện sắp xếp**



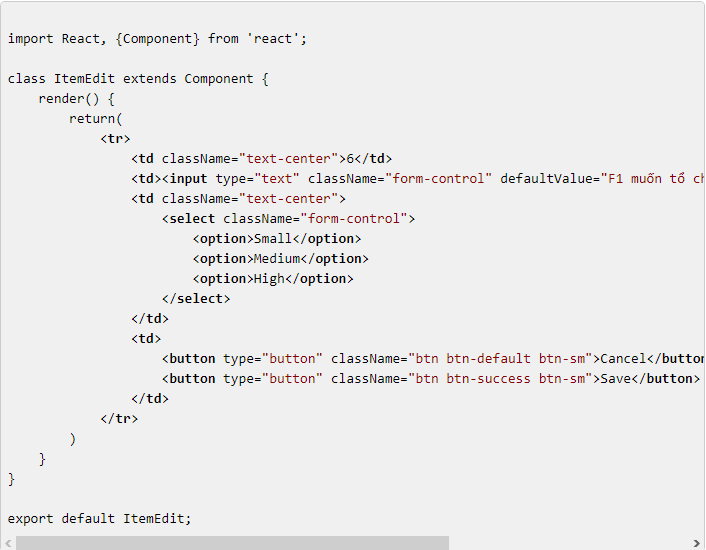
* **Tạo mới file Form.js để chứa form thêm**



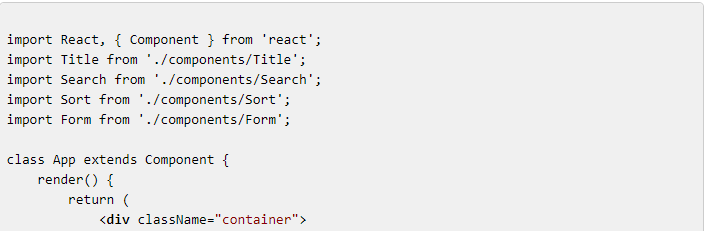
* **Tạo file tên là Item.js để chứa giao diện 1 hàng**



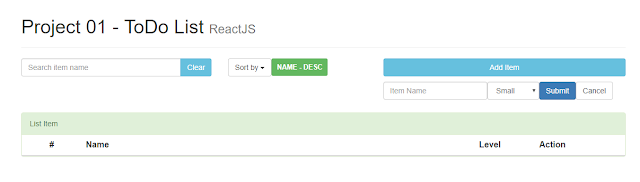
* **Tạo component ItemEdit để chứa giao diện sửa**



* **Sau khi đã tạo các component ta sẽ gọi chúng vào App.js và không gọi Item.js với ItemEdit vì chưa dùng.**

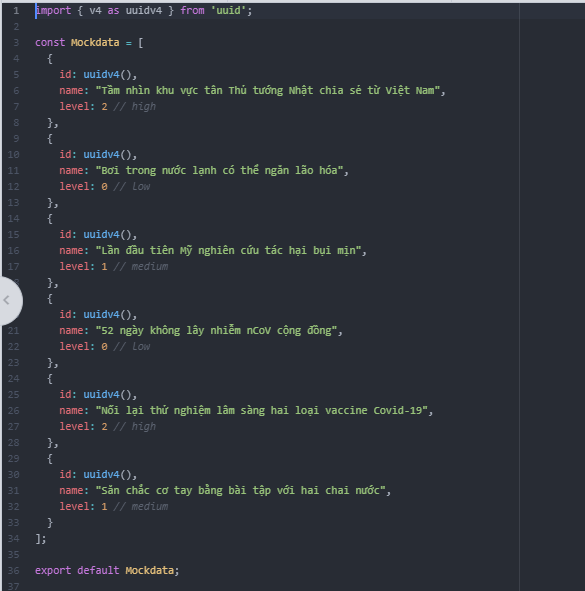


**Nếu làm tất cả thành công giao diện sẽ hiện**



1. **Tạo data tĩnh**

* **Vì là demo cơ bản nên ta làm việc với các dữ liệu có sẵn, vào src tạo thư mục mockdata. Tạo file trong thư mục đó tên Item.js**
* **Lưu ý là import đúng đường dẫn như hình thì không bị lỗi.**



* **Thêm package UUID**

**Mình sẽ sử dụng uuid để nó sinh mã cho id . Quay lại “Git Back Here” của project đang chạy nhấn Ctrl + C và nhấn Y để dừng chạy project**

**Hoặc có thể tắt mở Task Manager lên để End task Nodejs đang chạy.**

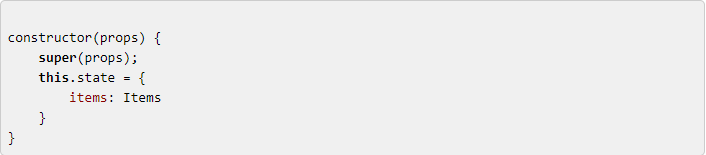
**Gõ lệnh “**npm install uuid --save**” nó sẽ tự sinh ra mã trong data cho mình**

* **Tiến hành Render() ở App.js**

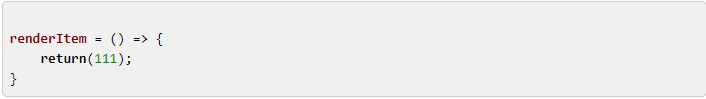
**Gọi các dữ liệu data vừa tạo**



* **Mình cho Items vào làm một state trong constructor của App.js**



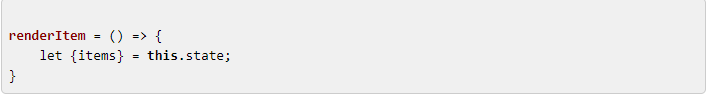
* **Mình viết thêm 1 hàm renderItem**



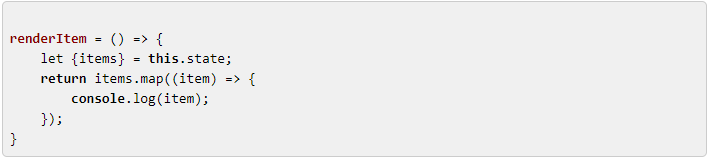
* **Gọi hàm này tbody**



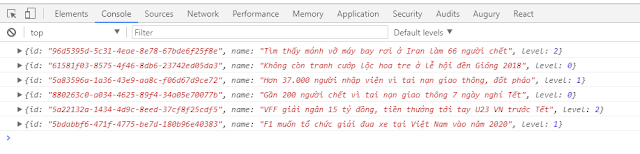
* **Mình lấy được state items chứa mảng dữ liệu**



* **Sử dụng hàm map để duyệt mảng items**



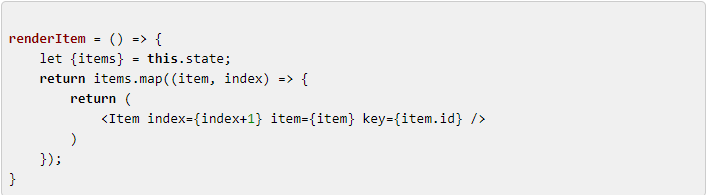
**Vào trình duyệt chạy project nhấn F12 chọn Console để thấy kết quả khi làm thành công map**



* **Để sử dụng các Item mình tạo thì phải gọi nó App.js**



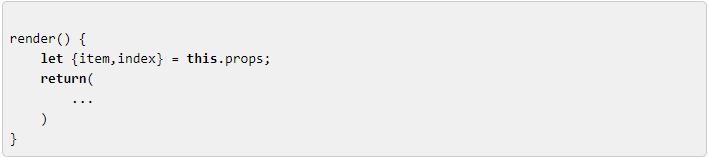
* **Thay đoạn console.log(item)**



* **In giá trị Item.js**

**Mình đã gửi các giá trị thông qua props <Item />**

**Nên giờ mở file Item.js trong folder component. Hàm render của nó phải lấy được các giá trị gửi quả từ App.js**



**Code file Item.js như sau**



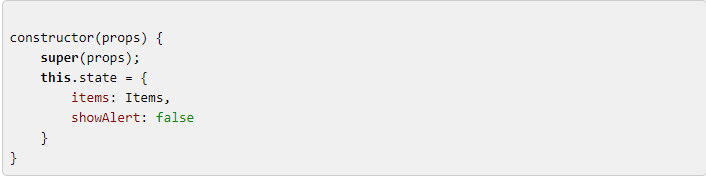


**Kết quả chúng ta thành công**



1. **Chức năng xóa**

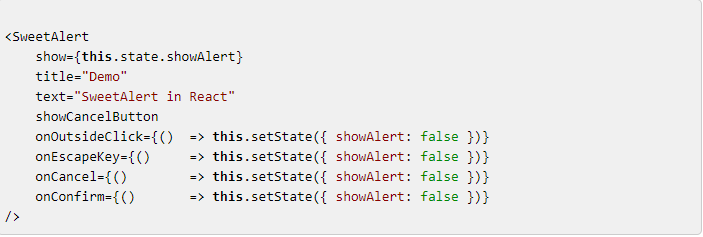
* **Mình sẽ sử dụng** SweetAlert dành cho React Js
* **Để thêm package ta gõ “npm install sweetalert-react”**
* **Việc ẩn hiện thì ban đầu là false vì không hiện ra ban đầu**



**Tiến hành gọi css của nó.**

**import './../node\_modules/sweetalert/dist/sweetalert.css';**

* **Sau khi đã css thành công ta cần**
* **Click vào cancel**
* **Nhấn ESC trên bàn phím**
* **Click vào vùng bên ngoài popup để ẩn**

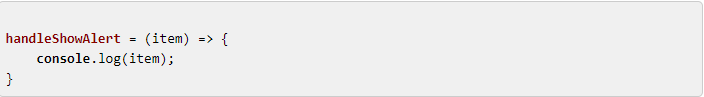


* **Lấy tên sản phẩm muốn xóa**

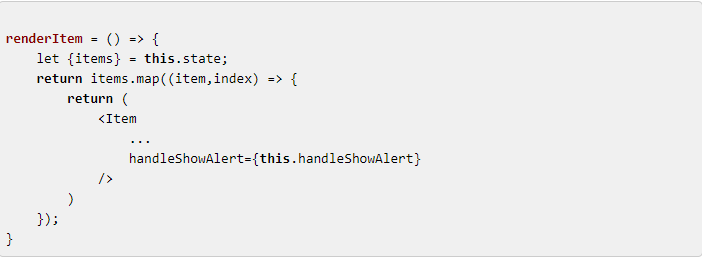
**Trong file App.js ta code như sau**



* **Mình sẽ viết hàm handleShowAlert để nhận vào item**



* **Gán hàm này vào vào làm props của component Item**

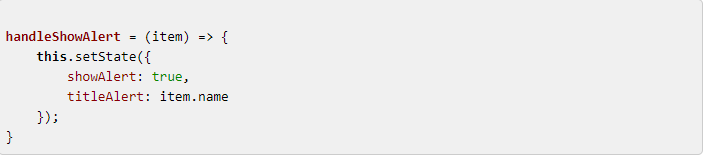


**Trong file Item.js viết sự kiên onClick cho nút Delete để truyền toàn bộ item**



**Vậy là xong chúng ta đã truyền được toàn bộ thông tin muốn xóa ra App.js**

**Trong App.js hàm handleShowAlert có nhiệm vụ popup và truyền đến item đó**



* **Lấy id của sản phẩm muốn xóa**

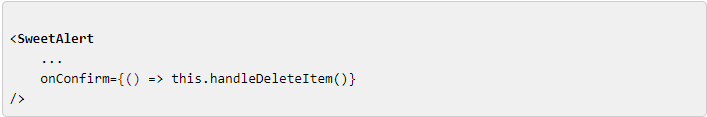
**Để xử lý việc xóa ta cần biết id của sản phẩm nên ta tạo một state lưu giá trị được gửi qua App.js**



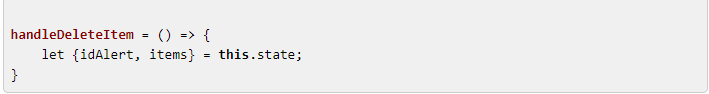
* **Xóa item**

**Chúng ta bấm vào sản phẩm sẽ có một popup hiện lên nhấn OK để xóa**

**Trong sự kiện onConfirm của Component SweetAlert gọi đến một hàm xử lý để xác nhận**

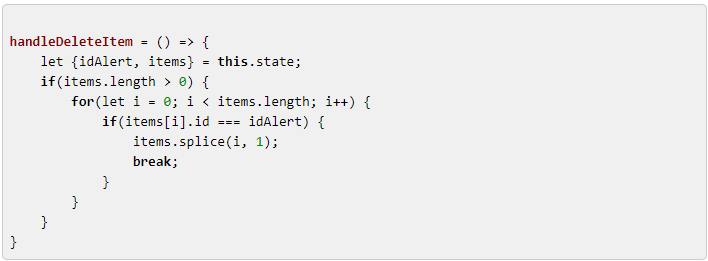


**Trong hàm handleDeleteItem mình lấy state chứa id muốn xóa và state chức mảng item ban đầu**

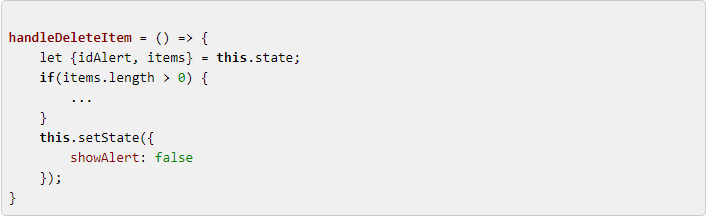


**Kiểm tra độ dài mảng > 0**

**Dùng hàm for để duyệt mảng ta so sánh có giống với id muốn xóa, nếu giống thì xóa**

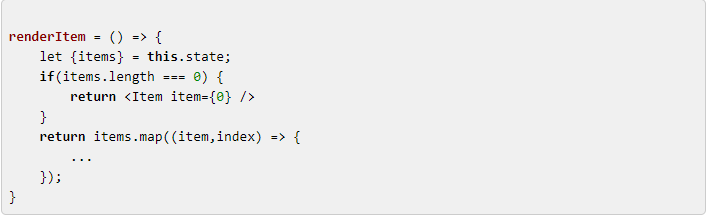


**Sau khi xóa xong thì ẩn popup đi bằng cách thay đổi giá trị state bằng false**



**Hiển thị thông báo khi không còn item nào**

**Trong hàm renderItem ta thêm lệnh kiểm tra độ dài của mảng**



**Trong file Item.js ta kiểm tra điều kiện và trả về một kiểm hiển thị riêng.**

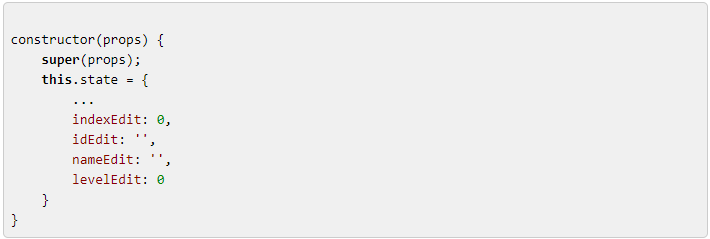


1. **Chức năng sửa item**

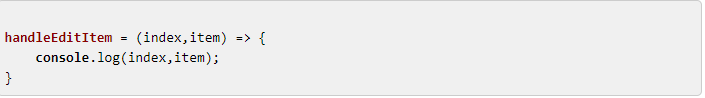
* **Lấy thông tin sản phẩm**

**Sau khi đã đọc qua phần xóa thì phần sửa này cũng hao hao như vậy**

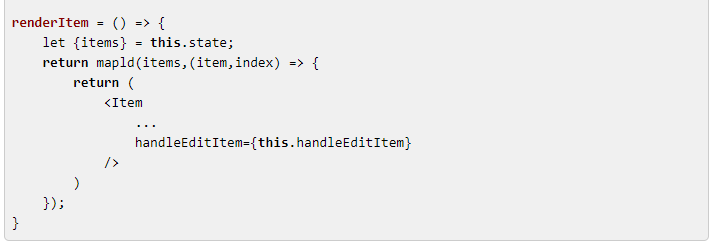
**Trong file App.js ta tạo state để lưu giá trị cần sửa.**



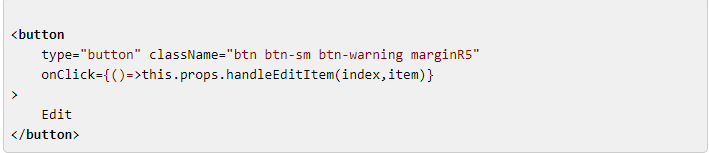
**Viết hàm nhận thông tin sửa**



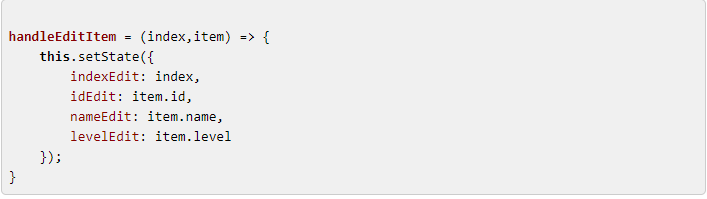
**Mình gắn trên hàm props của Component Item**



**Mở file Item.js viết hàm sự kiện cho clickEdit để gửi thông tin item muốn sửa**



**Trong file App.js giờ mình gắn các giá trị đó vào state để lưu**

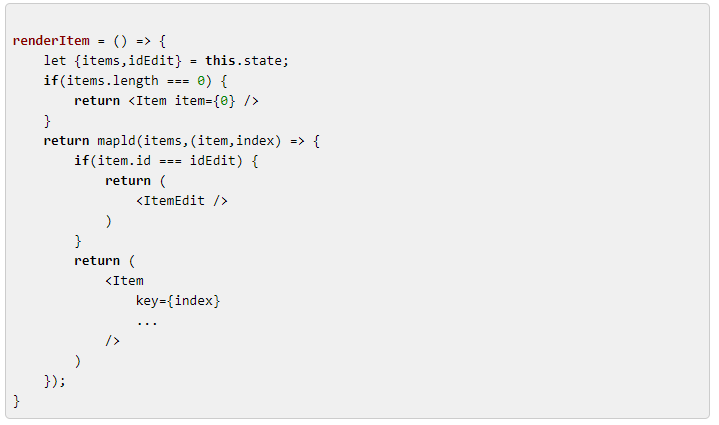


* **Sử dụng Component ItemEdit**

**Gọi thêm**

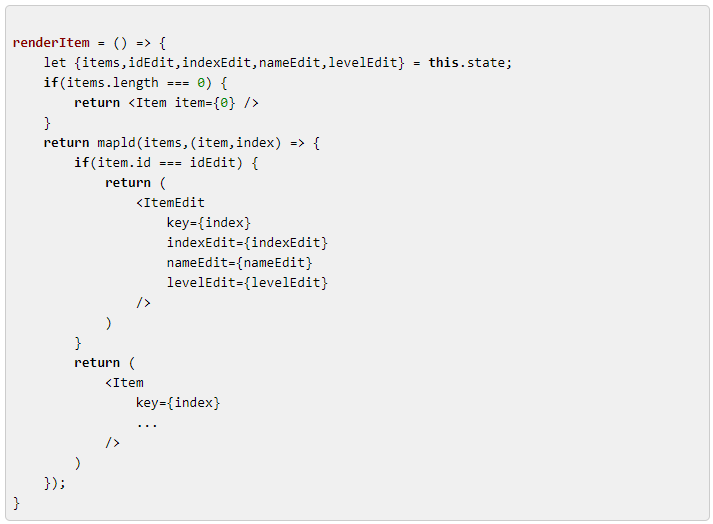


**Tại hàm renderItem ta lấy state id Edit**



* **Load số thứ tự và tên sản phẩm**

**Trong file App.js hàm renderItem ta lấy các giá trị state gửi vào ItemEdit**



**Tiếp tục chỉnh sửa trong file ItemEdit.js**



**Load tên item vào giá trị input**



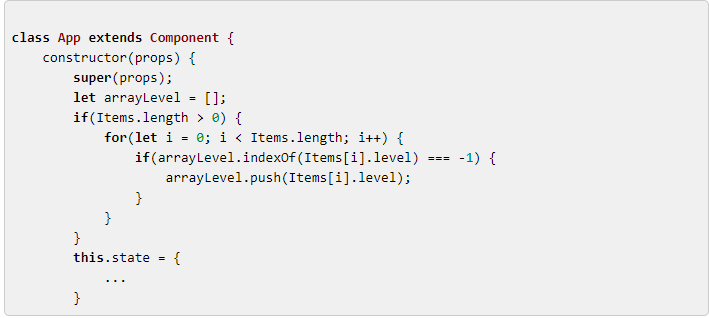
* **Load level sản phẩm**

**Trong project ta sẽ có 2 dạng hiển thị là project và opiton**

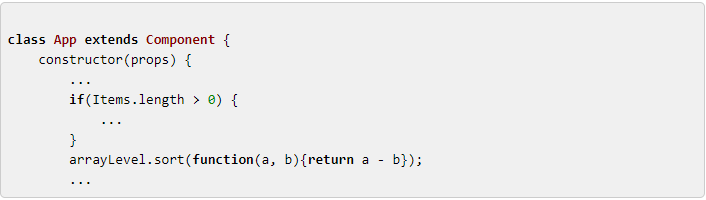
**Cách xử lí mảng:**

* **Tạo một mảng rỗng**
* **Duyệt mảng item và lọc lấy giá trị level**
* **Nếu level chưa có thì add vào mảng rỗng**
* **Nếu level có rồi thì không add nữa**

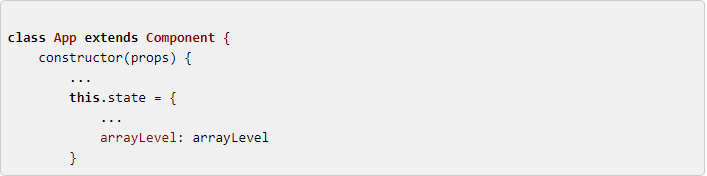
**Trong App.js tạo mảng sử dụng map để lọc giá trị level rồi push vào mảng đó**



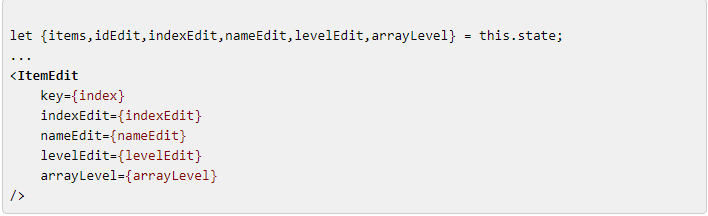
**Sắp xếp mảng này có giá trị từ nhỏ đến lớn**



**Mình đã có mảng Array lưu giá trị 0-1-2**

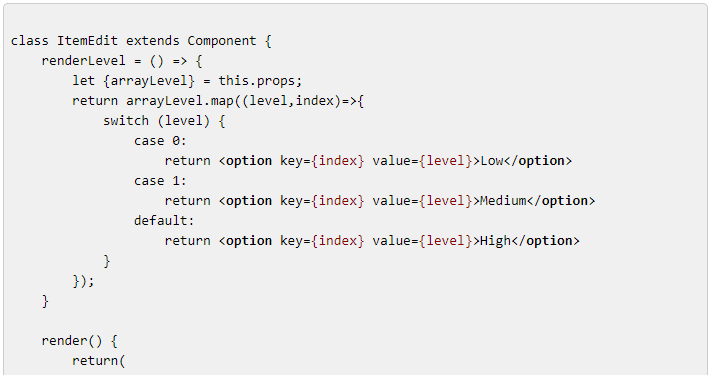


**Tại hàm RenderItem ta lấy state ArrayLevel gửi cho Component ItemEdit**



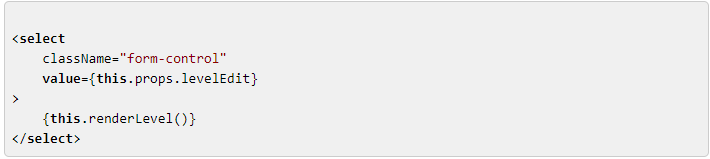
**Trong file ItemEdit.js sẽ code như sau để render các option tương ứng**

* **Lấy mảng Array**
* **Duyệt mảng Array level**
* **Với mỗi phần tử sẽ sinh ra một option tương ứng**



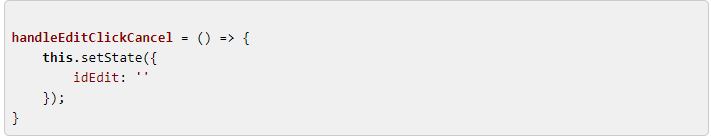


**Lấy giá trị mặc định hiển thị ra sau khi select**

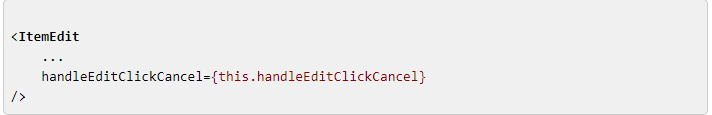


* **Sự kiện Cancel sửa item**

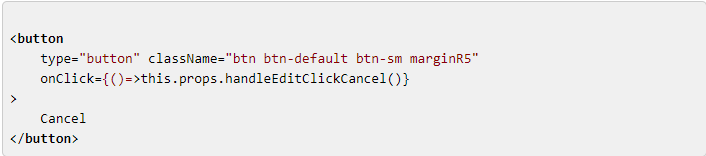
**Trong App.js ta viết hàm để set giá trị idEdit**



**Sau đó gắn hàm này vào Component ItemEdit**



**Trong file ItemEdit.js ta viết sự kiện onClick cho nút Cancel**

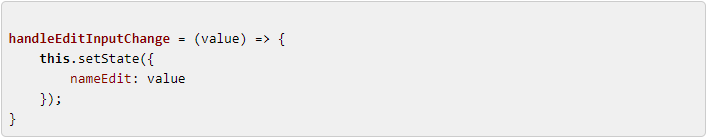


* **Viết hàm thay đổi input name**

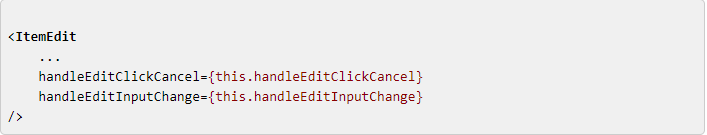
**Cách xử lí**

* **Viết một hàm thay đổi giá trị nameEdit**
* **Gửi qua cho Component ItemEdit**
* **Với sự kiện onChange gọi đến hàm đó**

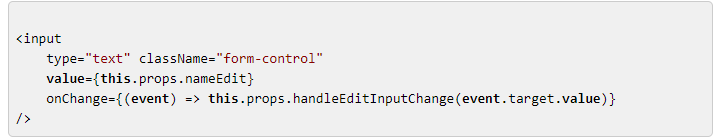
**Trong file App.js ta thay đổi giá trị state nameEdit và nhận tham số truyền vào**



**Sau đó gắn hàm này Component ItemEdit**

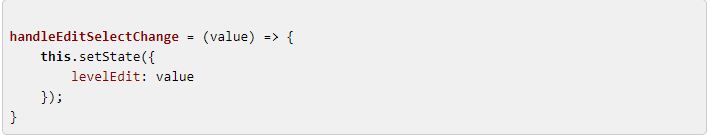


**Trong file ItemEdit.js viết sự kiện onChange cho input**

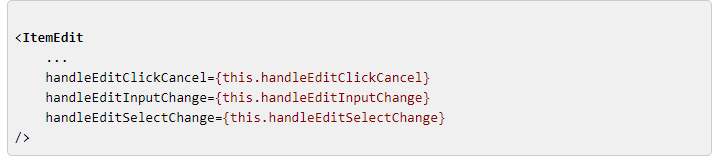


* **Hàm thay đổi select option**

**Trong file App.js ta viết hàm thay đổi state levelEdit**



**Sau đó gắn vào hàm Component ItemEdit**



**Trong file ItemEdit.js viết sự kiện onChange cho select**

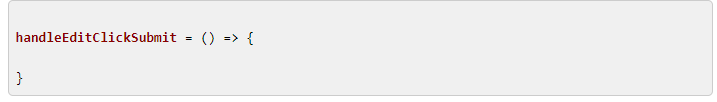


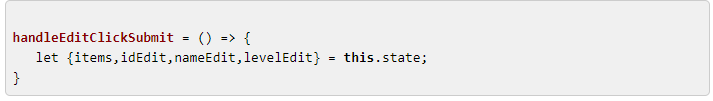
* **Viết sự kiện lưu khi sửa item xong**

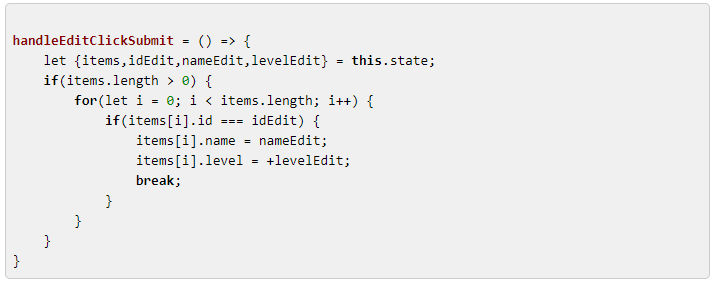
**Các bước làm:**

* **Lấy danh sách sản phẩm bao gồm id- tên- level**
* **Duyệt danh sách sản phẩm**
* **So sánh id mỗi item với item muôn sửa**
* **Thay đổi lại tên level của sản phẩm đó**

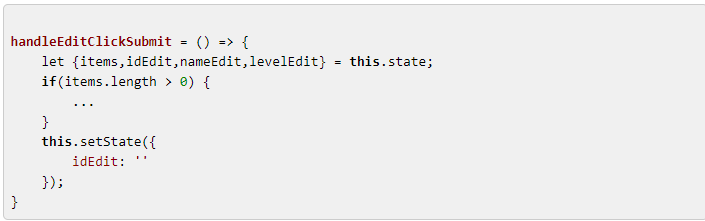
**Trong file App.js viết một hàm đê lưu khi sửa**



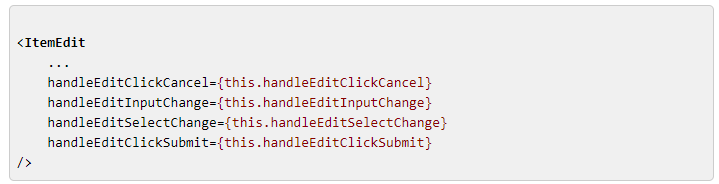




**Sau khi sửa , set lại edit về lúc ban đầu**



**Gửi hàm này qua Component ItemEdit**

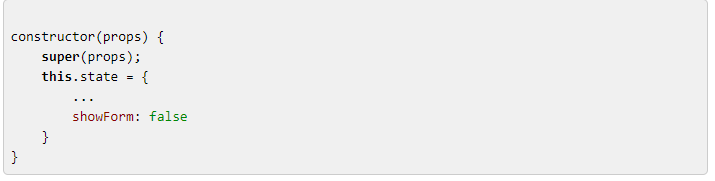


**Trong file ItemEdit.js ta viết sự kiện onClick cho nút Save**



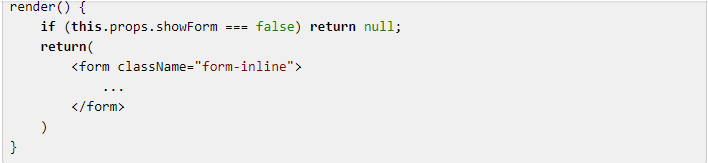
1. **Thêm Item**

* **Tại App.js ta khai báo một biến boolean để lưu việc ẩn form**

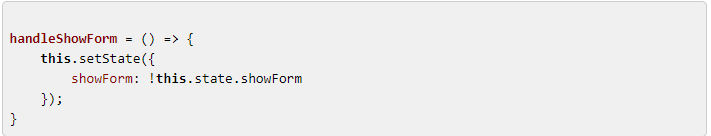


**Bên trong hàm render ta gửi Component Form xử lý**

 **Trong Form.js thì ta check giá trị được gửi sang**



**Quay lại App.js ta viết Toggle Form đảo giá trị**



**Kéo xuống hàm render ta gắn sự kiện onClick và thay đổi chữ cho phù hợp với Add Item và Close Item**

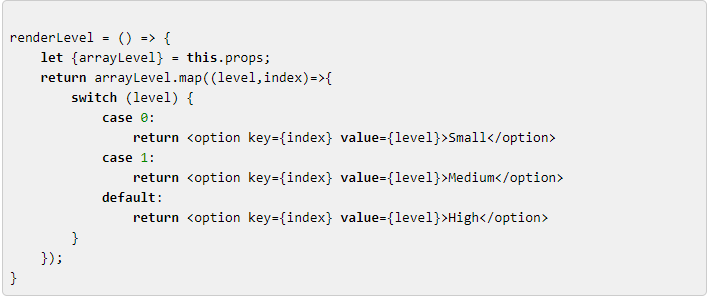


* **Viết hàm render level**

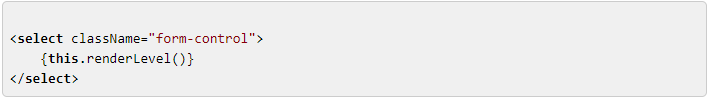
**Trong App.js tạo một mảng array level**



**Trong Form.js ta viết một hàm để nhận props arrayLevel, duyệt mảng và in ra thẻ option tương ứng**

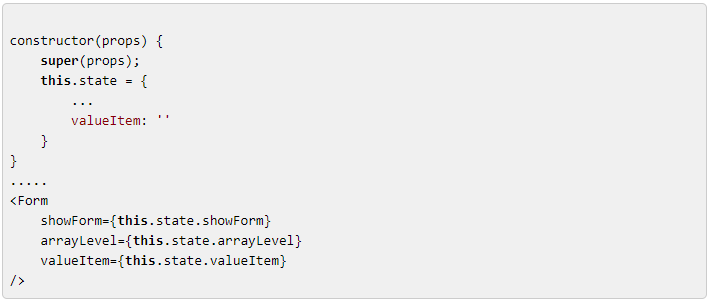


**Xóa các thẻ HTML option và thay bằng các hàm trên**

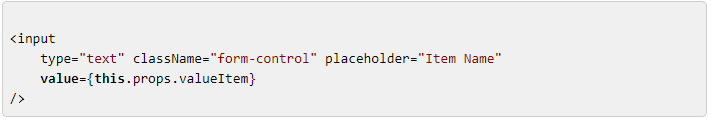


**-Viết hàm cho input onChange**

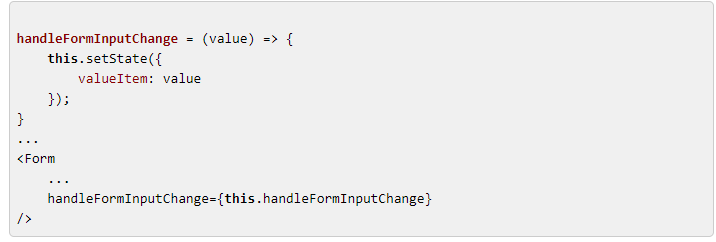
**Trong App.js ta sẽ tạo một kiểu string để lưu tên sản phẩm muốn add**



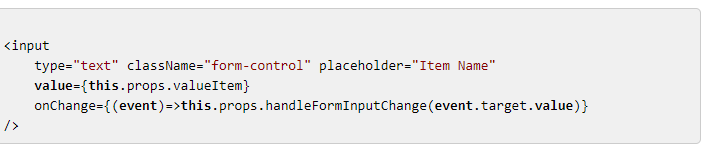
**Trong file Form.js ta gắn value băng props nhận được**



**Trong App.js ta viết một hàm thay đổi giá trị valueItem để nhận một tham số value đầu vào**

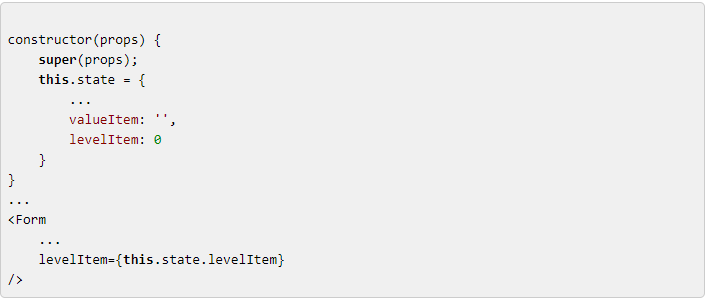


**Trong file Form.js ta viết sự kiện on Change đến props handleForminputChange**

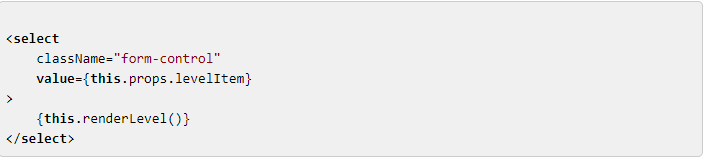


* **Viết hàm select- option onChange**

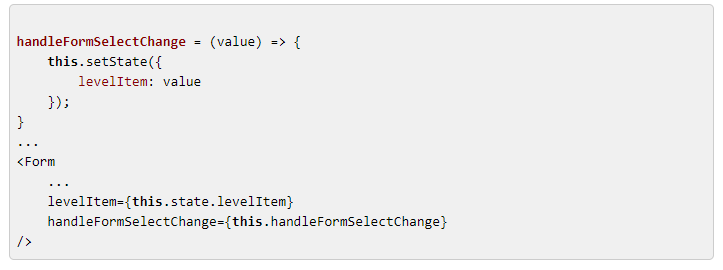
**Cũng tương tự như trên ta tạo một state kiểm number để lưu giá trị levvelItem, mặc định thì nó bằng 0**



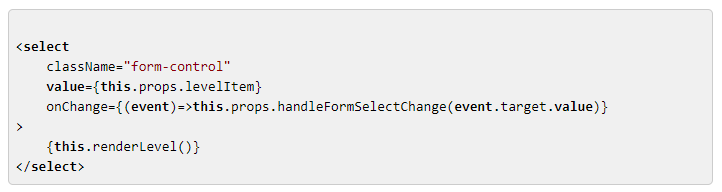
**Trong file Form.js ta gắn value bằng props nhận được**



**Quay lại App.js**

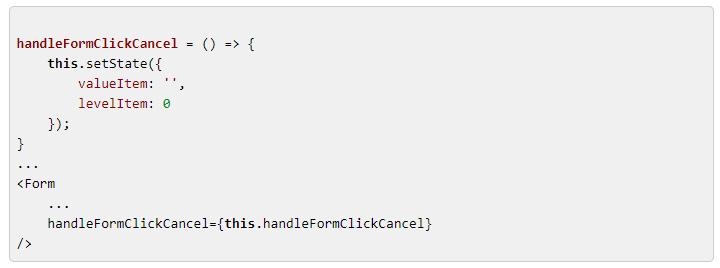


**Trong file Form.js ta viết sự kiện onChange**

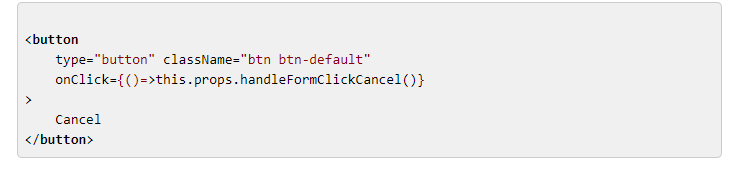


* **Viết hàm cho nút Cancel**

**Trong App.js ta viết hàm reset về giá trị mặc định**

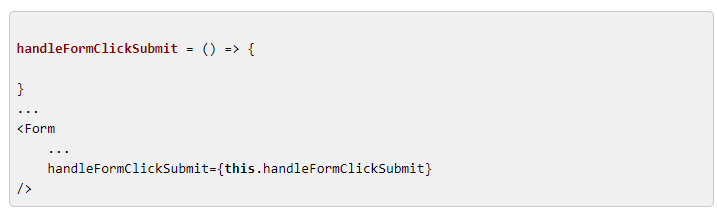


**Trong file Form.js ta gắn sự kiện cho nút Cancel**



* **Viết hàm cho nút Submit**

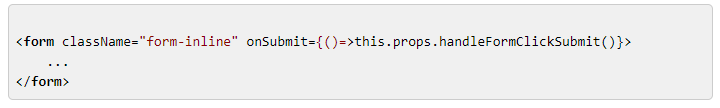
**Ta viết một hàm cho sự kiện ClickSubmit tại App.js**



**Trong Form.js ta gắn sự kiện onClick cho nút Submit**



**Gán luôn nút Enter khi người dùng không nhấn Submit**

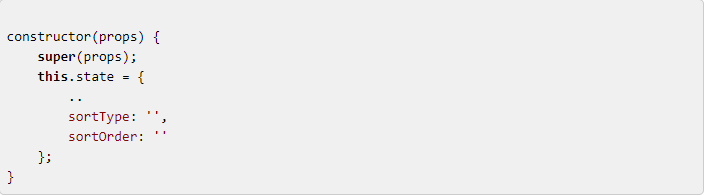


**Trong file App.js sau khi đã import uuid**

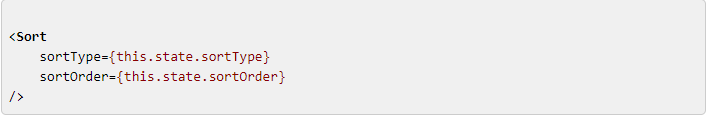


1. **Sắp xếp**

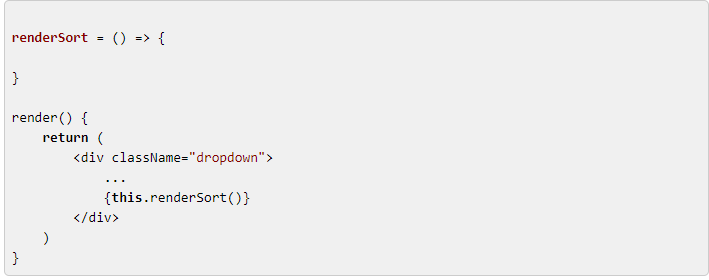
**Trong App.js mình thêm state vào là:**



**Ở phần render phía dưới ta chỉnh sửa lại**



**Do là đang là sử dụng kiểu dữ liệu tĩnh nên trong file Sorf.js mình viết như sau:**

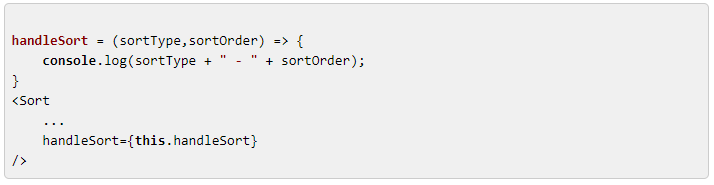


**Trong hàm renderSorf lấy hai giá trị rỗng , để check và render ra HTML tương ứng**

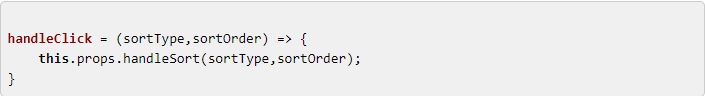


* + **Viết hàm thay đổi kiểu sắp xếp**

**Trong App.js viết thêm 2 hàm để thay đổi giá trị**



**Trong Sorf.js viết thêm hàm handle Click**

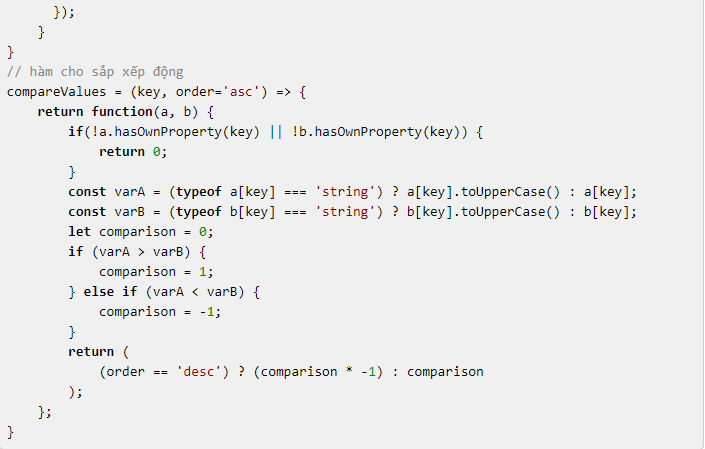


**Xuống phần render của Sorf.js**



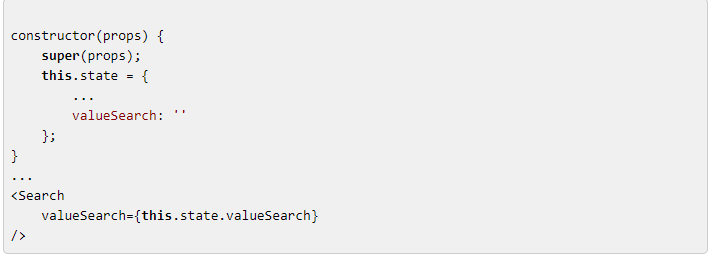
**Và trong file App.js ghi lại đầy đủ như sau, kiểm tra kĩ cho đúng với hình ảnh**



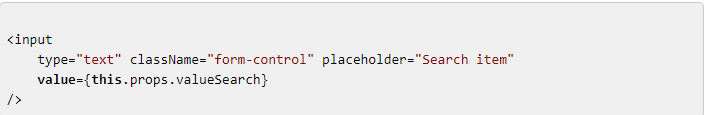


1. **Search item**

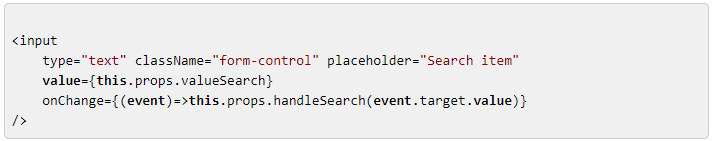
* **Trong App.js mình tạo kiểu dữ liệu String để lưu dữ liệu và gửi state này cho Component Search**



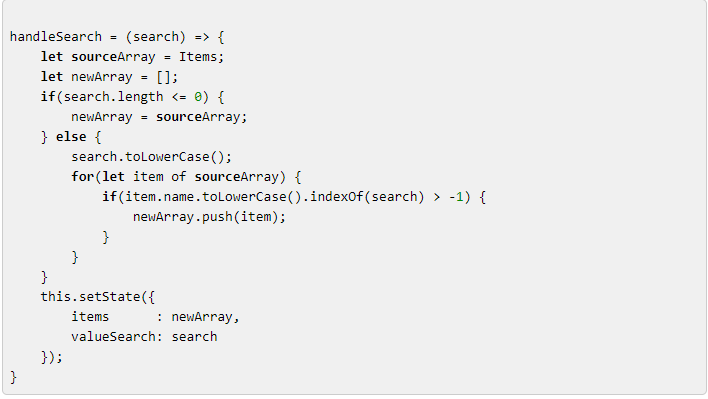
**Trong Search.js gắn giá trị value cho input bằng props nhận được**



**Viết hàm handle Search trong file App.js**



**Xử lí Search**

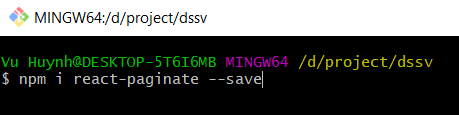


* **Viết hàm cho nút Clear**



1. **Phân trang**

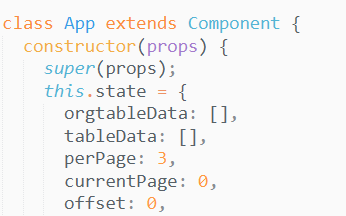
* **Mở Git Base trong project, cài React-Paginate cho project.**

****

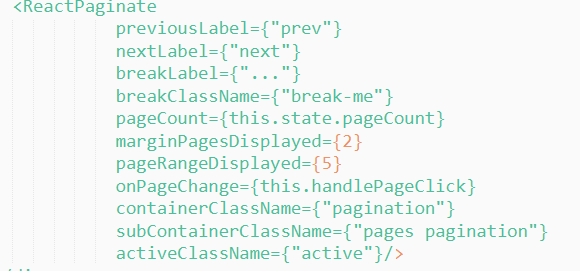
* **Chuyển đến thư mục src và mở tệp App.js**
* **Import ReactPaginate.**

****

* **Vào constructor method và khởi tạo.**

****

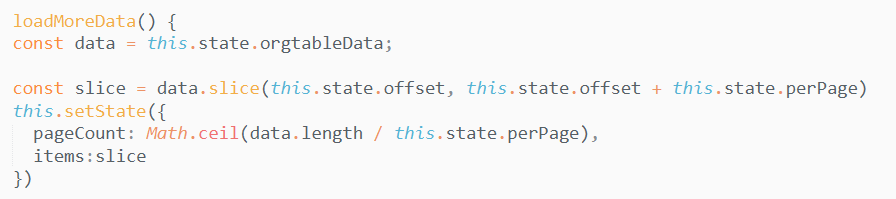
* **Vào Render và tạo Paginate.**

****

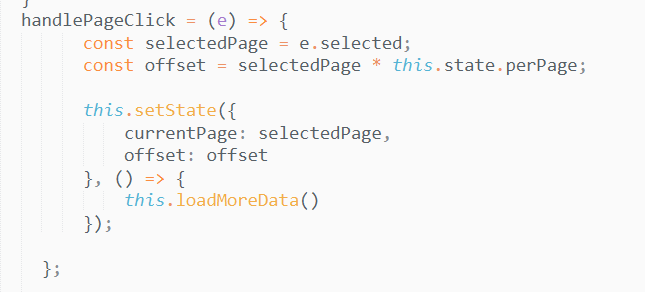
* **Viết hàm lấy pageCount và dữ liệu page đầu khi load.**

****

* **Viết hàm chia data cho từng trang.**

****

* **Viết hàm cho sự kiện PageClick.**

****