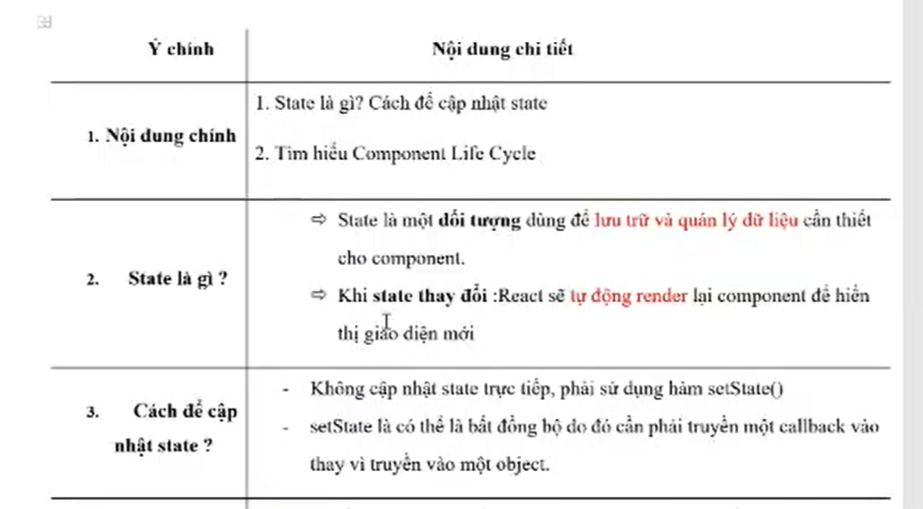
***Ghi chép state\_life-cycle***



**Vòng đời life-cycle:** khởi tạo

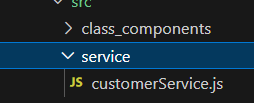
-> component dc gắn vào trong DOM(Mounting)

-> render dữ liệu

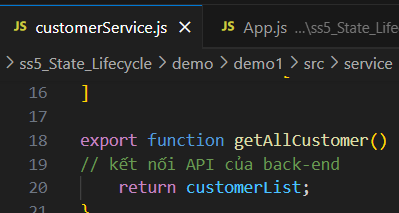
-> khi có sự kiện nào đó ( vd: onclick, hoặc một botton nào đó thay đổi cái state)

-> khi đó component dc render lại, khi nào component k dc sử dụng nữa thi nó sẽ về trạng thái Unmounting và k hiển thị nữa

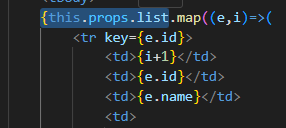
**Tạo file service để kết nối API( dữ liệu từ Backend)**

****

**Xây dựng phương thức để kết nối API**

****

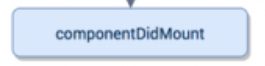
( khác file phải export)

* Sẽ k dùng props nữa
* Phải là 1 cái list call từ service lên

K sử dụng 

Vì rất dễ bị bất đồng bộ dẫn đến sai dữ liệu

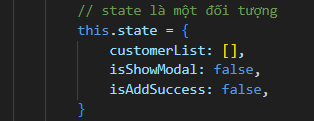
* Vì thế sử dụng componetDidMount để lấy render dữ liệu



* Nó sẽ render trc dc cái giao diện rồi sẽ quay lại hàm componentDidMount để lấy dữ liệu
* Và cho dữ liệu vào trong state
* Và khi dữ liệu nó thay đổi thì sẽ tự động render lại

Từ khóa this.state và this.props là từ khóa của nó rồi

Khởi tạo state( đối tượng lưu trữ dữ liệu)



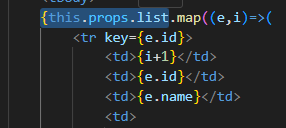
**Chương trình chạy: Khởi tạo(constructor) => wilmount(nhưng k có) => render dữ liệ**

* **didMount => rồi reset lại state -> state thay đổi thì component thay đổi**

Sử dụng state thay cho props

( đối tượng state lưu trữ danh sách khách hàng)

****

**Cũ:** 

Không thể cập nhất state trực tiếp dc( this.state=…)

* phải dùng setState()

**khi truyền vào đối số là 1 function khác thì là một callback**

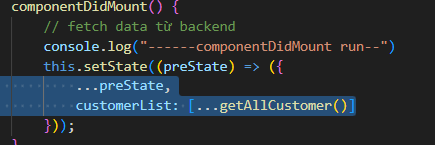
Thay vì truyền vào 1 object : ({customerList:getAllCustomer}) => k dc(có thể bị bất đồng bộ)

Thì truyền vào 1 function (arrow function)

* this.setState((preState) => (

**với preState là trạng thái trc khi nó cập nhật**

**cách cập nhật:** rải lại state ban đầu , và phần cập nhật



Là một mảng => cần rải lại 

Còn k sẽ là getAllCustomer()

Chú thích:

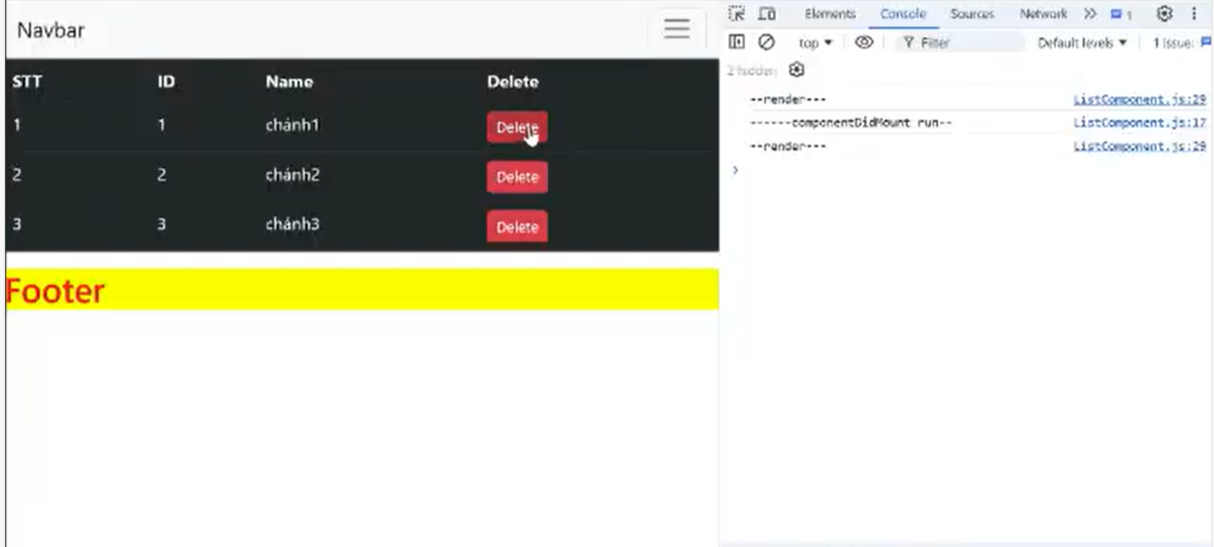
...preState,//rải lại các thuộc tính đã dc khởi tạo của state

customerList: [...getAllCustomer()] //(ghi đè) set lại các thuộc tính cần cập nhật

* **Sử dụng nhiều nhất 2 hàm:**

**-componentDidMount()**

**-componentDidUpdate():** khi mà trạng thái component thay đổi và muốn thực hiện tác vụ nào đó thì sẽ sử dụng hàm



**Tạo ấn delete thì mở hộp thoại modal:**

**B1**: tạo ra 1 function handleShowModal()

**B2:** khi onclick tại button sẽ **truyền vào 1 function** => mở hộp thoại

(nếu gọi đến function thì khi onclick r gọi đến sẽ bị render sai)

**Lưu ý :**

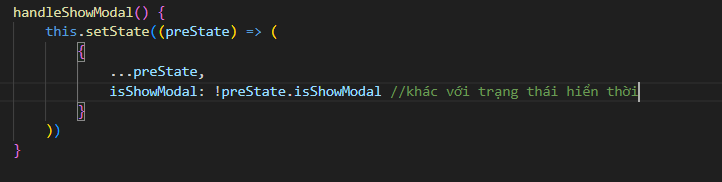
Truyền funcion là : vd onClick={this.handleShowModal} (k có ngoặc )

Gọi đến function là: vd onClick={this.handleShowModal()} (có ngoặc)

Thay đổi prop của deleteComponent của listComponent



Quy trình: khi bấm vào delete => thì gọi function handleShowModal => và function đó sẽ setState lại



Nếu chạy bị lỗi do : 

* Có 2 cách xử lý

**Cách 1:** chuyển thành arrow function:

={()=>{

this.handleShowModal();  
}

* + Thì mới dc gọi

**Cách 2:** vẫn sử dụng như cũ nhưng thêm hàm(ok hơn)

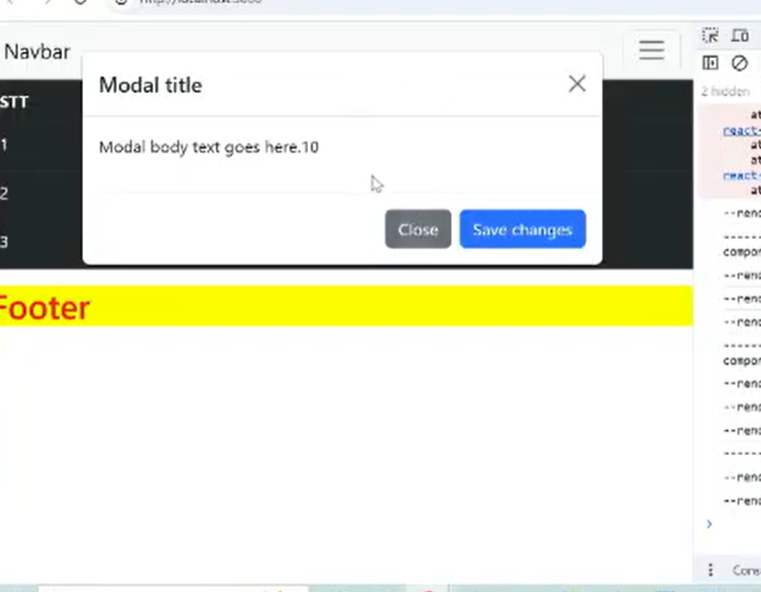
**.bind(this)**



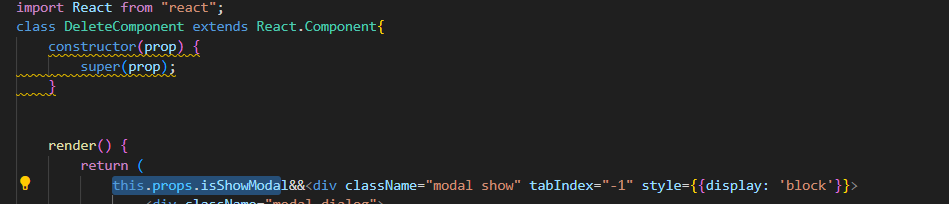
**Hoặc gán luôn ở trên và ở trong hàm giữ nguyên**



**Cải tiến nút tắt: (nút close)**

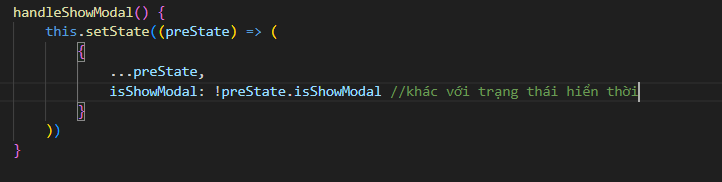
****

* **Khi state thay đổi -> tắt**



Thay đổi this.props.isShowModal (return có điều kiện)

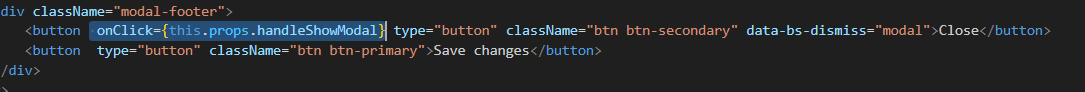
Prop có thể là một biến kiểu nguyên thủy, đối tượng , hàm

Sử dụng 

Nghĩa là ấn 1 lần nữa thì isShowModal sẽ thay đổi giá trị



( truyền function) (props)



Thay đổi ở mục modal

Quy trình: khi ấn nút close -> sẽ truyền đến 

Và sẽ gọi lại hàm modal lần nữa khiến cho nó nhận giá trị khác

**Phải tự làm nút xóa( khi ấn save -> xóa hẳn đi đối tượng)**

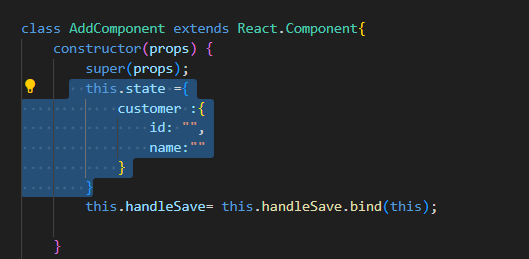
**Hướng dẫn: khi bấm vào xóa => có 1 function ở service thực hiện xóa và render dữ liệu lại**

**Thêm mới : (1 fille riêng) rồi import vào líst**

**B1: giao diện thêm**



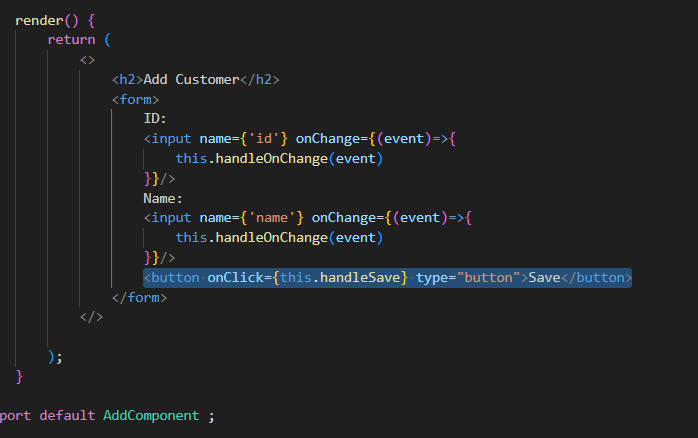
**B2: khởi tạo một state chứa đối tượng rỗng:**



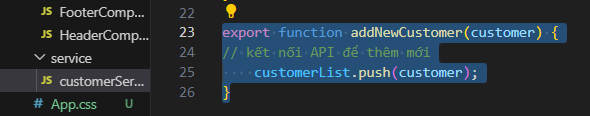
**B3: xây hàm save**



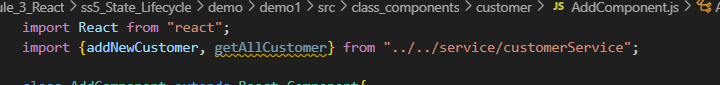
**B4: gọi đến hàm ở nút save**



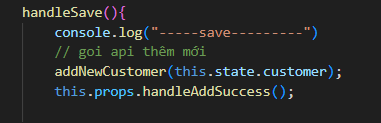
**B5: kết nối API để thêm mới( service)**

j

**b6: import hàm vào trong file Add..( thì mới sử dụng dc hàm trong file khác**

****

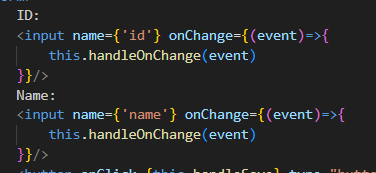
**B7: gọi API thêm mới trong save**



(nhưng dữ liệu nhập vẫn chưa thay đổi, customer khởi tạo vẫn là rỗng)

* Sử dụng hàm o**nchange**

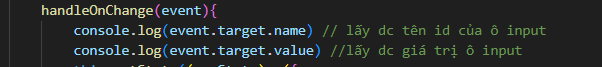
**B8:** sử dụng onChange trong input và gọi hàm handleOnChange => truyền 1 tham số vào:



( có thể ghi trùng biến gọi trùng hàm vì cùng trả về 1 sự kiện, ô khác trả về cái khac => ghi đè)

( ví dụ khi nhập id= 12 thì **event** sẽ nhận sự thay đổi**( giá trị)** của ô input mang **name=’id’ và value= 12**

* **Sẽ truyền lại hàm handle..**

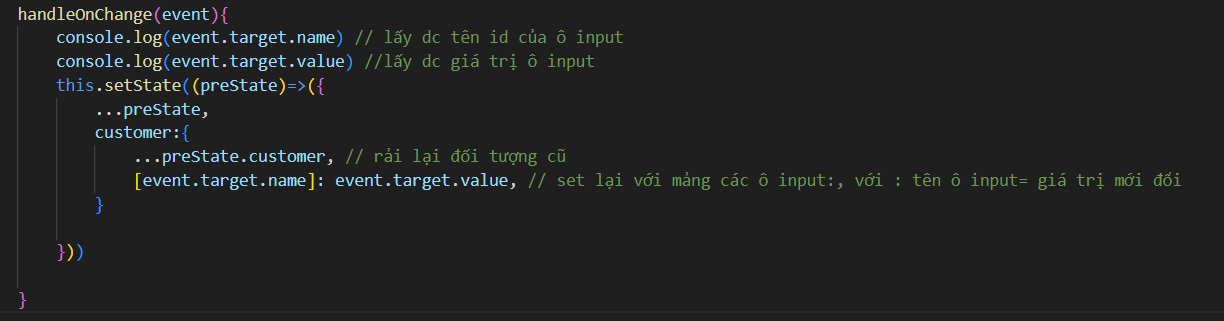
****

Cú pháp: event.target.name => lấy ra name của ô input

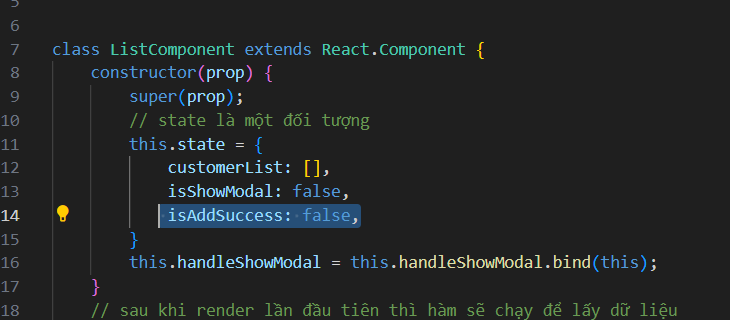
:event.target.value => lấy ra giá trị

( lấy biến event) : sự kiện khi onChange

* Xây function handleOnChange(event) cho thay đổi giá trị(setState)

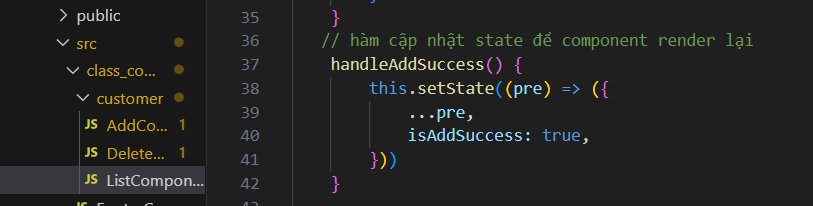


Tạo thuộc tính thêm của list( ở file list)

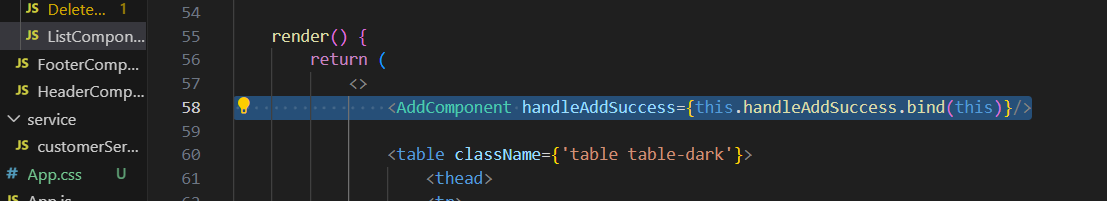


(Xác nhận thêm)

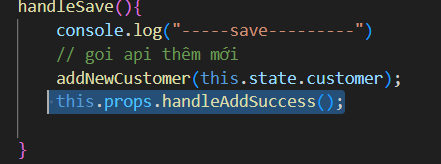
Xây hàm handleAddSuccess ở list ( để thay đổi list hiển thị)



Truyền vào file AddComponent props trọng thái của thêm



**B9**: gọi hàm handleAddSuccess ở hàm handSave()



**B10:** **khi hàm thay đổi => render lại dữ liệu của mảng nhưng chưa chuẩn**

* Sử dụng hàm componentDidUpdate

Nếu trạng thái thay đổi khác so với trạng thái đầu thì update

* 