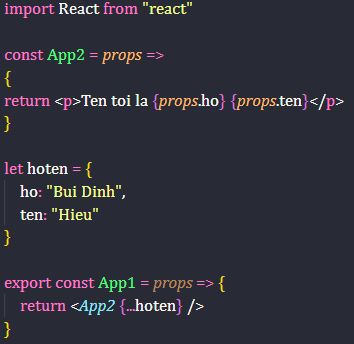
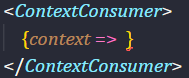
Trong bootstrap đa số (có thể là tất cả) các class có chiều rộng màn hình (sm, md, lg, xl) thì tính chất của nó là hoạt động khi màn hình lớn hơn hoặc bằng giá trị tương ứng. Ví dụ với class **px-md-3** thì nó hoạt động khi chiều rộng màn hình lớn hơn hoặc bằng 768px.

Ta có thể truyền props vào 1 component theo dạng đối tượng nhưng không truyền theo cách bình thường truyền đối tượng vào mà truyền theo cách như sau:

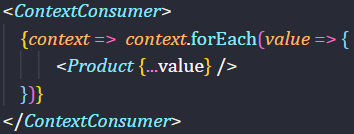


Đầu tiên ta phải khởi tạo đối tượng ở bên ngoài trước, sau đó sử dụng code **{…hoten}** để truyền các props vào component con đó. Khi đó chương trình sẽ hiểu các props **ho=“Bui Dinh” ten=“Hieu”**. Tuy nhiên nếu ta truyền vào bằng cách đưa cả đối tượng vào hoặc viết đối tượng ngay bên trong component thì gây ra lỗi.

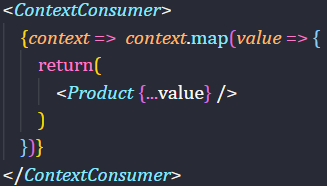
Khi tương tác dữ liệu sử dụng Consumer, giả sử có ví dụ sau:



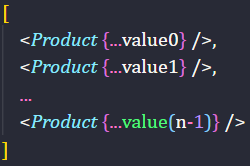
Phần đằng sau **context** phải là dữ liệu dạng mảng, dạng JSX, dạng chuỗi hoặc dạng số (nói chung nó là 1 dạng có giá trị) và trong context này ta không được dùng hàm forEach để xử lý dữ liệu vì thực chất hàm forEach là hàm xử lý dữ liệu chứ không phải hàm trả về giá trị dữ liệu. Ta có thể làm rõ hơn qua ví dụ sau:



Khi mảng context được xử lý qua forEach thì hàm thực thi các đoạn code bên dưới với lần lượt các phần tử trong mảng. Do là hàm thực thi code nên **<Product {…value} />** chẳng có ý nghĩa gì cho việc hiển thị nên chương trình báo lỗi và thay vào đó ta phải sử dụng hàm map do hàm này trả về giá trị mới. Thay forEach bằng map ta được:

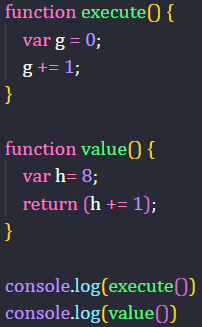


Chương trình đó tương đương với 1 mảng mới giả sử có tên là test bằng:



Đó chính là lí do vì sao hàm map lại thường được sử dụng trong JSX hơn forEach.

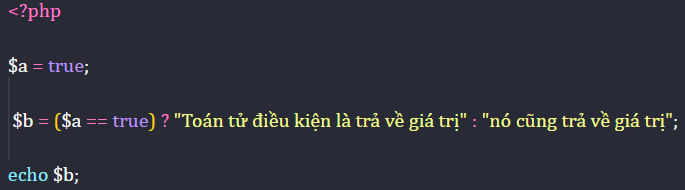
Ta có thể thấy rõ hơn sự khác nhau đối với hàm trả về giá trị và hàm thực thi như sau:



Ta thấy function execute() là hàm thực thi còn function value() là hàm trả về giá trị. Khi đó nếu ta console.log(execute()) thì chương trình gây ra lỗi còn console.log(value()) thì chương trình hiển thị ra kết quả bằng 9. Hàm trả về giá trị thì nó vừa thực thi vừa trả về giá trị còn hàm thực thi thì chỉ thực thi không thôi.

Đối với hàm pop và 1 số hàm xử lý mảng thì đều thuộc hàm trả về giá trị vì khi cho var a = arr.pop() thì a đều có giá trị.

Toán tử điều kiện trong cả JavaScript và PHP đều thuộc loại hàm trả về giá trị. Cụ thể như sau:



$b sẽ có giá trị là:



Để không bị nhầm lẫn khi truy cập các đường link trong reactjs thì route trang chủ ta nên để dấu “/”. Ví dụ:



Tránh để không có dấu “/” như sau:



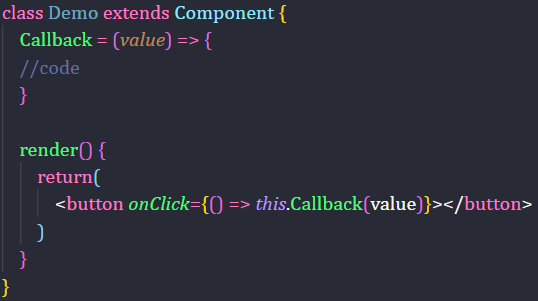
Bởi vì không có dấu “/” thì khi ta truy cập vào đường link bất kì ví dụ ta truy cập vào đường link **localhost:3000/detail** thì chương trình vẫn chỉ hiện ra component **Index** mặc dù ta đã **exact** route đó.

Trong 1 thẻ div được định dạng bởi class container khi ta chia grid trong container đó thì mỗi cột có giá trị bằng widthcontainer/12 và mỗi phần tử chiếm số col đó không bao gồm margin. Ví dụ 1 phần tử chiếm 2 col thì chiều rộng của nó là widthcontainer/6 và phần chiều rộng này không bao gồm margin của phần tử đó.

Khi sử dụng các hàm trong component được tạo bởi class, đối với những hàm arrow function (biểu thức hàm) thì ta không cần phải bind nó vào trong hàm constructor còn đối với hàm không phải arrow (khai báo hàm) thì cần phải bind.

Hàm callback là hàm được thực hiện khi có 1 sự kiện diễn ra và sự kiện đó mang 1 đặc trưng nhất định.

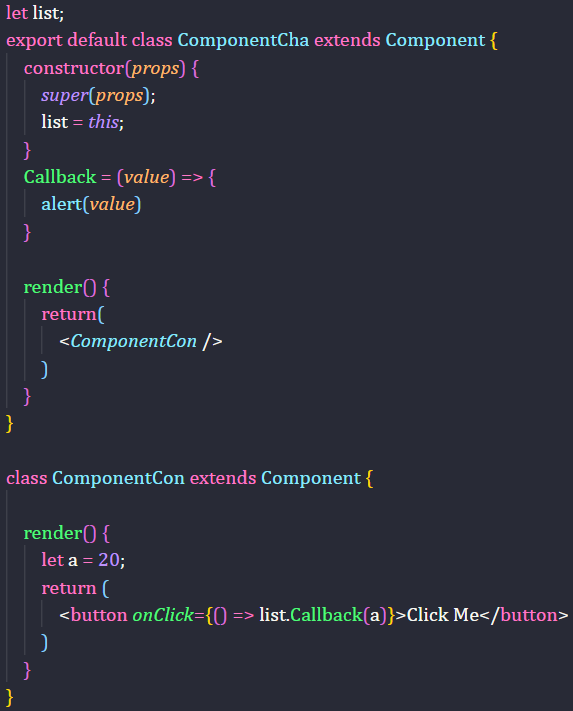
Thông thường hàm callback được sử dụng trong 1 component hoặc từ các component khác nhau nhưng có liên quan đến nhau. Ví dụ trong 1 component:



Khi khác component thì được truyền sử dụng hàm static như sau:



Hoặc ta có thể dùng gán đối tượng:



Trong JSX khi gọi hàm không sử dụng cặp dấu () ví dụ đúng:

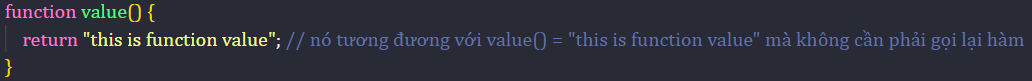


Ví dụ sai:

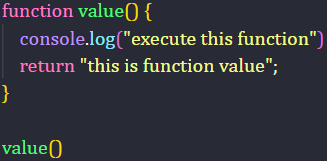


Hàm số gần giống như 1 biến, nó chỉ khác ở chỗ là nó có thể thực thi các dòng lệnh.

Khi thực thi 1 khu vực mà hàm số được định nghĩa trong đó thì hàm số đó nhận giá trị luôn chứ không cần phải gọi hàm thì nó mới nhận được giá trị. Ta có thể nhận ra qua biểu thức sau:

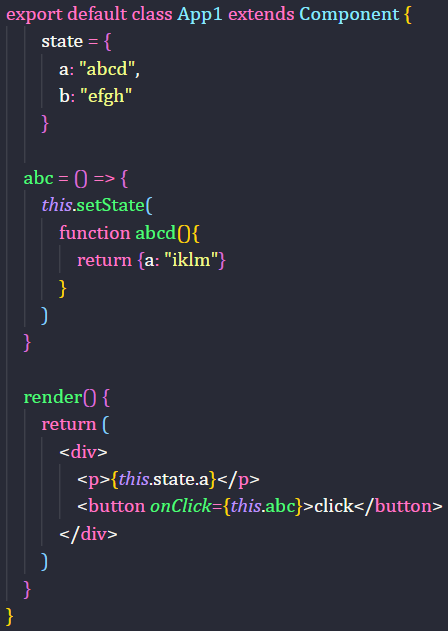


Qua đoạn code trên hàm số đã nhận giá trị luôn mà không cần phải gọi ra. Tuy nhiên nếu thực thi đoạn code bên trong hàm thì phải gọi ra thì phần thực thi bên trong mới được thực thi. Ví dụ:



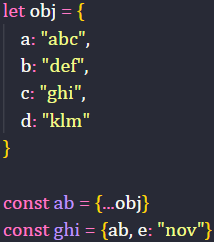
Khi chưa code **value()** dưới thì phần **console.log(“execute this function”)** chưa được thực thi và chưa đưa ra kết quả gì. Khi code đoạn code đó thì phần thực thi mới được thực thi. Cách thức hoạt động này tương tự như hàm vô danh.

Để làm rõ hơn sự khác biệt ta xét đoạn code:



Khi ta bấm vào button thì gọi đến hàm abc và thực thi hàm đó. Hàm đó thực thi this.setState và function abcd() bên trong được thực thi. Nếu function abcd() không đưa ra giá trị luôn  thì chương trình không chạy và không thay đổi được state của component App1 nhưng thực tế state.a của component App1 đã thay đổi.

Cho đoạn code sau:



Khi gh viết trong 1 đối tượng như thế kia thì hiểu ngầm rằng gh = gh : obj và kết quả **ghi** là đối tượng:

