**Tự học 10.3**

**Usememo:**

* Trả về giá trị đã được ghi nhớ
* Thông qua tạo hàm và một mảng các dependencies. useMemo chỉ tính toán các giá trị đã ghi nhớ khi một trong những dependencies thay đổi. sự tối ưu hóa này giúp tránh được những tính toán tốn kém trên mỗi lần render
* Nên viết code mà không có useMemo, sau đó mới dùng nó để tối ưu hóa hiệu suất.

**useCallback:**

* Trả về một giá trị đã ghi nhớ gọi lại
* thông qua một lệnh gọi nội tuyến và một mảng dependencies. nó trả về phiên bản đã được ghi nhớ của lệnh gọi lại và chỉ thay đổi khi một trong các dependencies thay đổi.
* useCallback(fn, deps) thì tương đương useMemo(() => fn, deps).

**Custom hook:**

* Là một hàm javascript bắt đầu bằng ‘use’ và có thể họi các hook khác
* Custom hook cho phép trích xuất logic thành phần thành các chức năng có thể sử dụng lại.
* Hook là các hàm nên có thể chuyển thông tin qua lại giữa chúng

**Error Bounderies:**

* Là các thành phần React bắt lỗi JavaScript ở bất kỳ đâu trong cây thành phần con của chúng, ghi lại các lỗi đó và hiển thị giao diện người dùng dự phòng thay vì cây thành phần bị lỗi.
* Error boundaries không bắt lỗi cho:
* Trình xử lý sự kiện
* Mã bất đồng bộ
* Kết xuất phía máy chủ
* Lỗi được tạo ra do chính nó( không phải con của nó)
* Một class trở thành error boundary nếu nó xác định một trong hai (hoặc cả hai) phương thức static getDerivedStateFromError () hoặc componentDidCatch (). Sử dụng static getDerivedStateFromError () để hiển thị một giao diện người dùng dự phòng sau khi một lỗi được đưa ra. Sử dụng componentDidCatch () để ghi thông tin lỗi.
* Error boundaries chỉ bắt lỗi trong các thành phần bên trong chúng. nó không thể tự bắt lỗi. Nếu một error boundary không cố gắng hiển thị thông báo lỗi, lỗi sẽ lan truyền đến ranh giới lỗi gần nhất phía trên nó. Điều này cũng tương tự như cách khối catch {} hoạt động trong JavaScript.