# **UseState**

## **Tại sao dùng callback function trong useState?**

Nếu **giá trị ban đầu** cần phải tính toán phức tạp, ví dụ:

* Đọc dữ liệu từ **bộ nhớ (local storage)**.
* Tính toán một số lớn hoặc dữ liệu nặng.

Nếu bạn **không dùng callback**, React sẽ chạy tính toán này **mỗi lần render lại**, làm chậm ứng dụng.

Callback function (() => {...}) sẽ chỉ chạy một lần khi component được hiển thị lần đầu.

Khi render lại (do state hoặc props thay đổi), hàm sẽ không chạy lại => Tối ưu hiệu suất.

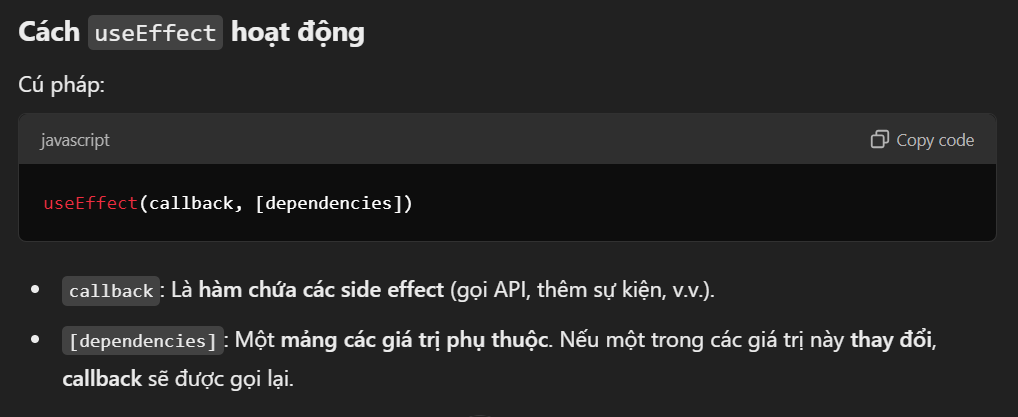
## **Tình huống cần dùng callback trong useState**

Giả sử bạn cần **tính toán phức tạp** hoặc **lấy dữ liệu từ API** để tạo **giá trị ban đầu** cho state. Bạn **chỉ muốn thực hiện tính toán hoặc gọi API một lần duy nhất** (khi component được hiển thị lần đầu) để tránh lãng phí tài nguyên.



# **UseEffect**

useEffect là một **hook** trong React và React Native dùng để **thực hiện side effect** (những tác vụ ngoài, như gọi API, cập nhật DOM, hoặc thiết lập bộ đếm thời gian). Nó được gọi **sau khi component render**, giúp đồng bộ hóa logic của bạn với giao diện.



Có 3 loai:

**useEffect**(callback)

**useEffect**(callback, [])

**useEffect**(callback, [deps])

# Custom hook

Custom Hook là một hàm trong React mà bạn tự định nghĩa để chứa logic và state có thể tái sử dụng.

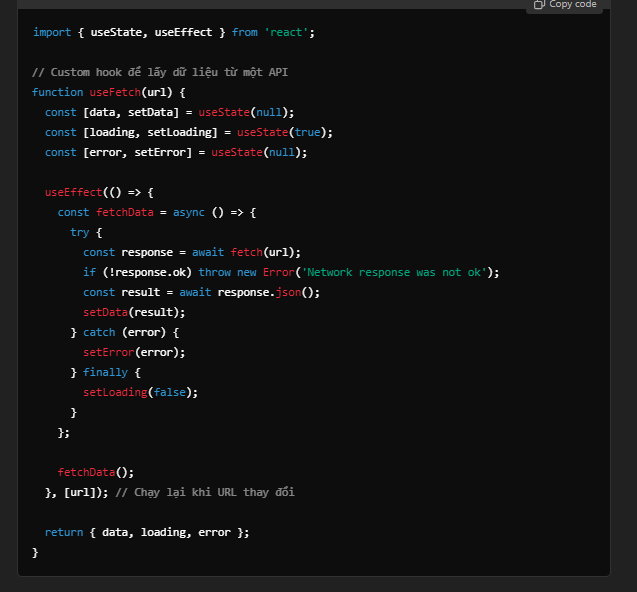
Nó cho phép bạn tách biệt logic phức tạp ra khỏi các component, giúp mã của bạn sạch hơn và dễ bảo trì hơn.

**Tại sao sử dụng Custom Hook?**

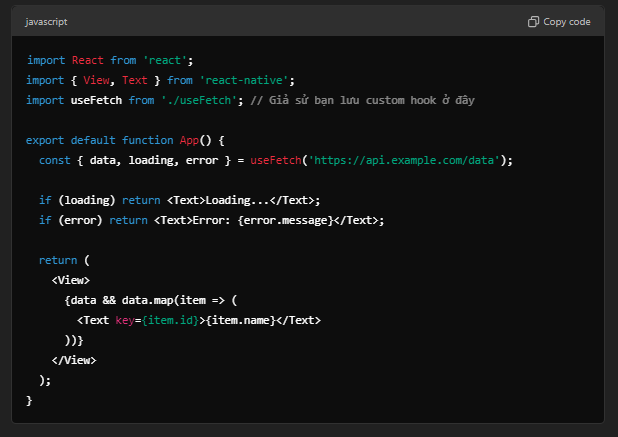
1. **Tái sử dụng mã**: Bạn có thể sử dụng cùng một custom hook trong nhiều component khác nhau mà không cần phải viết lại mã.
2. **Giảm độ phức tạp**: Bằng cách tách logic phức tạp ra khỏi component, bạn có thể làm cho component trở nên đơn giản và dễ hiểu hơn.
3. **Tổ chức mã tốt hơn**: Giúp bạn tổ chức mã của mình theo cách mà các logic có liên quan được nhóm lại với nhau.

**Cách tạo Custom Hook**

Một custom hook thường được đặt tên bắt đầu bằng từ "use" để phân biệt với các hàm thông thường



**Cách sử dụng Custom Hook**

****