**Vì sao nên dùng Redux?**

Redux giúp quản lý **state tập trung** thay vì để mỗi component tự quản lý. Điều này đặc biệt hữu ích khi ứng dụng React có nhiều tầng component, cần chia sẻ dữ liệu xuyên suốt.  
Nếu không có Redux, bạn sẽ phải truyền props lòng vòng (prop drilling), rất rối rắm.

Khi dùng Redux, ta có:

* Một “store” duy nhất lưu toàn bộ state.
* Các “action” để yêu cầu thay đổi state.
* Các “reducer” xử lý action và cập nhật state.
* Các component chỉ việc **connect** vào store để lấy state hoặc dispatch action, không cần truyền props vòng vo.

**3 bài tập ví dụ**

1. **Todo List**:
   * Không Redux: mỗi component TodoItem tự quản lý checked/un-checked.
   * Với Redux: danh sách todo lưu tập trung, khi thêm/xóa/sửa thì mọi component liên quan update đồng bộ.
2. **Counter App nhiều nút**:
   * Không Redux: mỗi CounterButton có state riêng → đếm lệch nhau.
   * Với Redux: một state count trong store, mọi nút đều dispatch action INCREMENT/DECREMENT → hiển thị đồng nhất.
3. **E-commerce Cart**:
   * Không Redux: dữ liệu giỏ hàng nằm ở component cha, truyền props xuống con → code rối.
   * Với Redux: store lưu cartItems, component ProductList, Header (icon giỏ hàng), và CartPage đều đọc từ cùng một store → lúc thêm sản phẩm, mọi nơi tự update.

**Khi nào áp dụng Redux thì tối ưu?**

* **Ứng dụng lớn**, nhiều component chia sẻ chung state (auth, giỏ hàng, user profile, theme).
* Cần **dự đoán được logic**: vì Redux quy định rõ action → reducer → state.
* Muốn **debug dễ dàng**: Redux DevTools cho phép xem lại lịch sử action.

**Khi không áp dụng Redux thì tối ưu hơn?**

* Ứng dụng nhỏ, ít state dùng lại (ví dụ form login).
* Dùng Context API là đủ, không cần thêm Redux làm phức tạp.
* Khi bạn chỉ có state cục bộ, thay đổi ở một chỗ duy nhất.