Tuần 1: React Cơ Bản

JSX: (JavaScript XML) là một phần mở rộng cú pháp của JavaScript cho phép bạn viết mã trông giống như HTML trong JavaScript, thường được sử dụng với React.

Component:

Component là cách React tạo nên bố cục của một website. Giống như trò chơi xếp hình, thay vì chúng ta code các khối HTML riêng lẻ, thì ở đây, chúng ta sử dụng Component

Component khác HTML ở chỗ là nó sử dụng cú pháp JSX. Với JSX, chúng ta có thể code logic Javascript cùng với HTML

- Function Component.

- Class Component.

Component State

Là Javascript Object

Miêu tả trạng thái (state) hiện tại của Component: data/UI-state

State của Component có thể được cập nhật, ví dụ như: đóng/mở Modal...

Props

Props là viết tắt của từ ‘property’.

Props là một javascript object (giống State, cũng là một object)

Dịch nghĩa là ‘tài sản’ vì props sinh ra để giúp component con có thể kế thừa lại (sử dụng được) ‘tài sản’ component cha để lại (ở đây là truyền data từ cha xuống con)

Lifecycle : chu kỳ sống và hoạt động của 1 thành phần trong quá trình thực thi của ứng dụng .

Mỗi thành phần trong React đều có vòng đời mà bạn có thể theo dõi và thao tác trong ba giai đoạn chính của nó.

- **Ba giai đoạn là: Mounting, Updating, and Unmounting.**

+ Mounting: có nghĩa là đưa các phần tử vào DOM.

constructor():phương thức này được gọi trước bất kỳ thứ gì khác, khi thành phần được khởi tạo và đó là nơi tự nhiên để thiết lập trạng thái ban đầu và các giá trị ban đầu khác.

getDerivedStateFromProps():

render():phương thức này là bắt buộc và là phương thức thực sự xuất HTML sang DOM.

componentDidMount(): phương thức được gọi sau khi thành phần được hiển thị.

+ Updating

getDerivedStateFromProps()

shouldComponentUpdate()

render()

getSnapshotBeforeUpdate()

componentDidUpdate()

+ Unmounting

componentWillUnmount()

State Management :Một chiếc app sẽ có rất nhiều State, đôi khi chúng hoạt động độc lập, hoặc phụ thuộc cũng như là ràng buộc lẫn nhau nên chúng ta phải quản lý nó. Đó chính là State Management.

Tuần 2:

UseMemo()

- Là một react hooks giúp mình tạo ra một memoized value và chỉ tính toán ra value mới khi dependencies thay đổi.

> Nhận vào 2 tham số: 1 là function, 2 là dependencies.

> Return memoized value

> Chỉ tính toán value mới khi dependencies thay đổi.

> Nếu dùng empty dependencies thì không bao giờ tính toán lại value mới.

Cú pháp: const variableName = useMemo(callback, dependency);

Trong đó:

- callback: là một hàm được gọi lại, lần render đầu tiên luôn chạy vào hàm này.
- dependency: là sự phụ thuộc, khi dependency thay đổi thì useMemo cập nhật lại giá trị.

UseCallback()

- Là một react hooks giúp mình tạo ra một memoized callback và chỉ tạo ra callback mới khi dependencies thay đổi.

> Nhận vào 2 tham số: 1 là function, 2 là dependencies.

> Return memoized callback.

> Chỉ tạo ra function mới khi dependencies thay đổi.

> Nếu dùng empty dependencies thì không bao giờ tạo ra function mới.

Cú pháp: const functionName = useCallback(callback, dependency);

Trong đó:

- callback: là một hàm được gọi lại, lần render đầu tiên luôn chạy vào hàm này.
- dependency: là sự phụ thuộc, khi dependency thay đổi thì useCallback mới tạo ra một hàm mới.

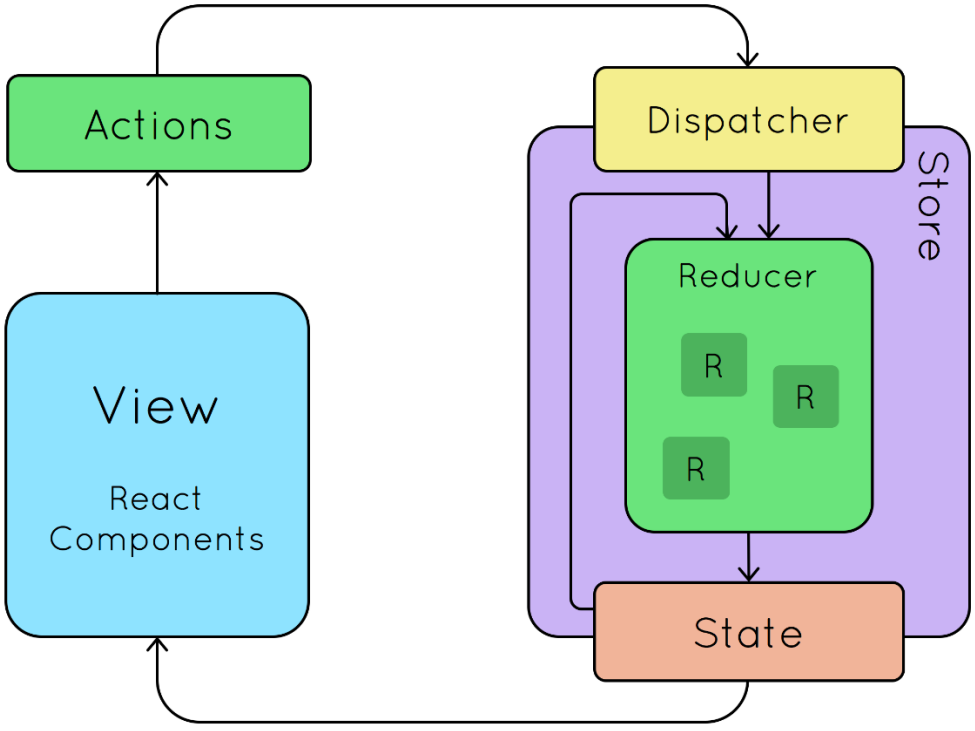
- Formik & Yup

+ **Formik :** là một thành phần giúp bạn xây dựng biểu mẫu. Nó sử dụng mẫu đạo cụ kết xuất phổ biến bởi các thư viện như React Motion và React Router.

+ **Yup:** là một trình tạo schema (lược đồ) dùng để phân tích và xác thực giá trị khi chạy (runtime). Bạn có thể định nghĩa một lược đồ, biến đổi một giá trị để phù hợp với lược đồ đó, kiểm tra cấu trúc của một giá trị hiện có, hoặc làm cả hai.

Lược đồ của Yup rất linh hoạt và cho phép mô hình hóa các kiểm tra phức tạp, liên quan lẫn nhau hoặc chuyển đổi giá trị.

- Redux: Redux là 1 thư viện Javascript để quản lý state của ứng dụng, thường được sử dụng với javascript framework như React. Cơ chế hoạt động của nó được tóm gọn trong 1 sơ đồ :



Redux Toolkit: là một thư viện giúp mình viết Redux tốt hơn, dễ hơn và đơn giản hơn. (tiêu chuẩn để viết Redux)

RTK bao gồm :

configureStore()

- Có sẵn Redux DevTools
- Có sẵn redux-thunk để thực hiện async actions

createReducer()

createAction()

- React Router: là thư viện giúp quản lý định tuyến trong ứng dụng web sử dụng React.

Các component chính của React Router:

- **BrowserRouter:** Bọc ứng dụng để kết nối với URL của trình duyệt.
- **Routes:** Bọc danh sách các Route để điều hướng các component.
- **Route:** Định nghĩa tuyến đường đến component cụ thể.
- **Link:** Chuyển đổi giữa các URL mà không tải lại trang (tương tự thẻ `<a>`).
- **Outlet:** Xác định vị trí hiển thị component trong route (tương tự `{props.children}`).
- **NavLink:** Giống như Link nhưng thêm class `active` nếu URL trùng khớp.
- **Navigate:** Tự động chuyển hướng đến một trang khác.

Nested Router: Tạo các route con lồng nhau.

Index Routes: Hiển thị component của route con bên trong route cha.

Dynamic Routes: Tạo các route động với tham số.

Hooks của React Router:

- **useParams:** Lấy tham số từ URL.
- **useNavigate:** Điều hướng đến đường dẫn khác hoặc quay lại trang trước.

Protected Routes:
Quản lý truy cập vào các trang công khai (public) và riêng tư (private). Nếu người dùng chưa đăng nhập, họ chỉ có thể truy cập trang public và bị chuyển hướng sang trang đăng nhập khi cố truy cập private.