Ministry of SMEs and Startups

KOSME Korea SMEs and Startups Agency

LIKELION

JOBKOREA

Theory 14
Global State Management in NextJS

WEEK 04

Nội dung chính

Global State

- **Global State** là trạng thái được chia sẻ giữa nhiều component không có quan hệ trực tiếp với nhau.
- Trong **Next.js**, state có thể nằm ở:
  - Component state (local)
  - Global state (Zustand, Redux, RTK Query)
  - Server state (SSR, getServerSideProps)

Global State

❑ **Ví dụ:**

```
// local state
const [count, setCount] = useState(0);

// global state (Zustand / Redux)
const { selectedItem } = useStore();
```

Global State

❑ Khi nào cần Global State?

Tình huống	Có nên dùng global state?
Quản lý user đăng nhập	Có
Modal hiển thị ở nhiều nơi	Có
Form dữ liệu tạm giữa nhiều bước	Có
Toggle UI trong 1 component	Không

Zustand

- **Zustand**: thư viện quản lý **Global UI/Local State**
- **Ưu điểm:**
  - Gọn nhẹ, đơn giản
  - Không cần boilerplate
  - Không phụ thuộc Redux

Zustand

- **Ví dụ store quản lý modal và selected item:**
  - Khởi tạo **store/modalStore.js**:

```
export const useModalStore = create((set)) => ({
    isOpen: false,
    selectedItem: null,
    openModal: (item) => set({ isOpen: true, selectedItem: item }),
    closeModal: () => set({ isOpen: false, selectedItem: null }),
}));
```

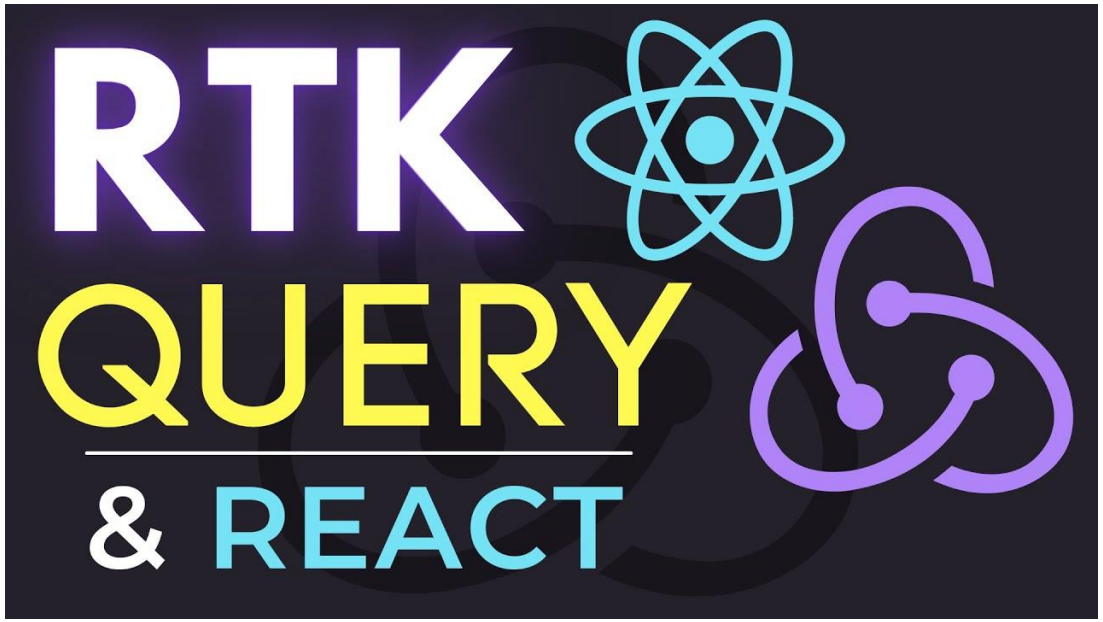
Zustand

- **Ví dụ store quản lý modal và selected item:**
  - Sử dụng **modalStore** trong **component**:

```
const { isOpen, selectedItem, openModal } = useModalStore();
```

RTK Query

- **RTK Query**: thư viện quản lý **Global Server State**
- **Ưu điểm:**
  - Tự động caching, re-fetch
  - Quản lý loading/error/success state
  - Tích hợp Mutation và invalidation



RTK Query

- **Ví dụ tạo các hàm gọi article API theo các endpoint cụ thể:**
  - Khởi tạo **features/articleApi.js**

```
export const articleApi = createApi({
    baseQuery: fetchBaseQuery({ baseUrl: '/api' }),
    endpoints: (builder) => ({
        getArticles: builder.query({ query: () => 'articles' }),
        createArticles: builder.mutation({ … }),
    }),
});
```

RTK Query

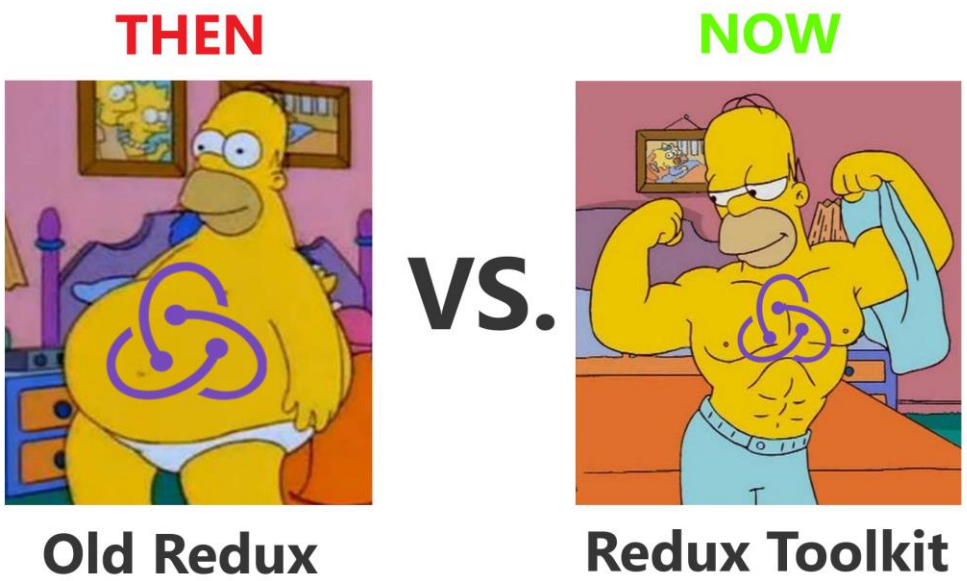
- **Ví dụ tạo các hàm gọi article API theo các endpoint cụ thể:**
  - Khởi tạo **features/articleApi.js**

```
export const articleApi = createApi({
    baseQuery: fetchBaseQuery({ baseUrl: '/api' }),
    endpoints: (builder) => ({
        getArticles: builder.query({ query: () => 'articles' }),
        createArticles: builder.mutation({ … }),
    }),
});
```

RTK Query

- **Ví dụ tạo các hàm gọi article API theo các endpoint cụ thể:**
  - Sử dụng **articleApi** trong **component**:

```
const { data, isLoading } = useGetArticlesQuery();
```



Zustand vs RTK Query

Tiêu chí	Zustand	RTK Query
Loại state quản lý	UI, local app	Dữ liệu từ Server
Cần setup Redux?	Không	Có
Có hỗ trợ caching?	Không	Có sẵn
Dễ tích hợp với modal/UI	Tuyệt vời	Không phù hợp

Zustand vs RTK Query

- Có thể sử dụng độc lập hoặc sử dụng kết hợp **Zustand** và **RTK Query.**
- Ví dụ sử dụng kết hợp **Zustand, RTK Query** vào các chức năng:
  - Danh sách bài viết (RTK Query)
  - Modal thêm/sửa bài viết (Zustand)
  - Xóa bài viết với mutation (RTK Query)

Tóm tắt bài học

- ❑ Global State
- ❑ Zustand
- ❑ RTK Query
- ❑ Zustand Vs RTK Query