**Tìm hiểu redux-toolkit phần 1**

[Trending](https://viblo.asia/trending)

**Giới thiệu**

**Redux** là công cụ quản lý **state** phổ biến và khá gây tranh cãi trong hệ sinh thái **react**, vì có khá nhiều **dev** than phiền là **redux** dài dòng và khó sử dụng. Riêng mình thì không thấy khó nhưng nhiều lúc cũng thấy hơi dài dòng thật . Vì để tạo ra một **store** hoàn chỉnh thì chúng ta cần qua khá nhiều bước, tạo nhiều **file** với các đoạn **code** lặp lại khá nhiều, ngoài ra **redux** không hề xây dựng một quy chuẩn rõ ràng cho việc viết **logic.**, cũng như ngoài **redux** thì đôi khi ta cần cài đặt thêm một số thư viện nữa như **middleware, selectors**... thì mới có thể viết hoàn chỉnh **logic** quản lý **state**.  
**redux-toolkit** là một **package** được sinh ra nhằm giải quyết phần lớn những vấn đề kể trên, được phát triển bởi chính chủ **reduxjs team**, giúp chúng ta viết **code redux** nhanh gọn, hoàn chỉnh theo một quy chuẩn thống nhất.  
Trong bài viết này mình sẽ giới thiệu về **redux-toolkit**, bài viết sẽ dành cho những bạn đã sử dụng hay biết về **redux**, còn với những bạn chưa từng sử dụng **redux**, mình nghĩ rằng các bạn nên tiếp cận với **redux** cơ bản trước, nắm bắt được cách thức hoạt động cũng như các khái niệm chính như **actions, reducer, store...**

**Cài đặt và tạo một store**

Cài đặt **redux-toolkit**

# NPM

npm install @reduxjs/toolkit

# Yarn

yarn add @reduxjs/toolkit

Tạo một **store** trong **redux**

// store.js

import { createStore } from 'redux'

const INCREASEMENT = 'increasement'

const DECREASEMENT = 'decreasement'

// actions

export const increasement = () => {{ type: increasement })

export const decreasement = () => ({ type: decreasement })

const initialState = { count: 0 }

function rootReducer(state = initialState, action) {

switch (action.type) {

case INCREASEMENT:

return { count: state.count + 1 };

break;

case DECREASEMENT:

return { count: state.count - 1 };

break;

default:

return state

}

}

const store = createStore(rootReducer)

export default store

Chúng ta vừa khởi tạo một **store** cơ bản nhất bằng **redux**, với chỉ một **state** duy nhất (**count**) và lắng nghe chỉ 2 **action type** để thực hiện tăng hoặc giảm giá trị **count**. Mặc dù vậy ta vẫn thấy có một chút hơi phức tạp phải không, dù chỉ là một **store** đơn giản, ví dụ như việc khai báo **types**, hơi dài dòng một chút nhỉ, và giá trị này cũng chỉ dùng khi tạo **action** và trong **reducer**.

const INCREASEMENT = 'increasement'

const DECREASEMENT = 'decreasement'

Ngoài ra chúng ta phải tự xử lý **logic** bên trong **reducer** bằng hoàn toàn bằng **js** mà không có một quy chuẩn nào rõ ràng, ví dụ như **action** không nhất thiết phải trả về **type**, **reducer** thực ra không nhất thiết phải sử dụng **switch** để lắng nghe **action**.

**redux-toolkit: configureStore**

Chúng ta bắt đầu với hàm **configureStore** của **redux-toolkit**. Như ví dụ ở trên, chúng ta khởi tạo một **store** bằng hàm **createStore** của **redux** với tham số nhận vào là một **reducer**, **configureStore** sẽ làm điều tương tự như vậy, chúng ta sẽ khởi tạo **store** theo phương pháp sau:

import { configureStore } from '@reduxjs/toolkit';

//...

const store = configureStore({

reducer: rootReducer

})

Về cú pháp thì không khác nhau nhiều phải không? Tuy nhiên thay vì chỉ khởi tạo một **store** đơn thuần, **configureStore** sẽ mặc định thiết lập cho phép sử dụng **redux devtool** để **debug** và theo dõi quá trình cập nhật **state** cũng như thiết lập sẵn một số **middleware**.

**redux-toolkit: createAction**

Tiếp theo chúng ta sẽ tìm hiểu cách tạo một **action** với hàm **createAction**, nhận vào tham số là một **string** và trả về một hàm khởi tạo **action** với **type** sẽ là giá trị **string** được truyền vào

import { createAction } from '@reduxjs/toolkit';

const increment = createAction('INCREMENT')

console.log(incrementNew())

// {type: "INCREMENT"}

Đầu tiên thì **createAction** giúp ta khởi tạo **action** một cách ngắn gọn hơn với việc tạo **action** bằng **javascript function** đơn thuần. Ngoài ra ở ví dụ trên mình không hề khai báo trước giá trị **type** của **action** mà truyền trực tiếp **string** vào làm tham số, vì **createAction** cung cấp sẵn khả năng truy xuất **type** của **action** rồi, lại tiết kiệm thêm vài dòng **code** nữa  , cú pháp sẽ như sau

import { createAction } from '@reduxjs/toolkit';

const increment = createAction('INCREMENT')

console.log(increment.toString())

// "INCREMENT"

console.log(increment.type)

// "INCREMENT"

**redux-toolkit: createReducer**

Một hàm nữa mình muốn giới thiệu trong bài viết này, đơn giản là chúng ta sẽ không phải tạo **reducer** hoàn toàn bằng **js** thuần nữa.  
**createReducer** cho phép chúng ta khởi tạo **reducer** một cách đơn giản và trực quan hơn khá nhiều so với cách viết **thuần**, **logic** nhìn sẽ trực quan hơn như một **object tham chiếu**, với mỗi **key** là một **type** và **value** là một **reducer function**. Không dài dòng nữa, cùng xem cú pháp nhé:

import {createReducer} from '@reduxjs/toolkit'

const increment = createAction('INCREMENT')

const decrement = createAction('DECREMENT')

const initialState = { count: 0 }

const counter = createReducer({ count: 0 }, {

[increment.type]: state => ({ count: state.count + 1 }),

[decrement.type]: state => ({ count: state.count - 1 })

}

**createReducer** nhận vào 2 tham số là giá trị ban đầu của **state** và **object** giúp **reducer** lắng nghe và cập nhật **state**, nếu ở ví dụ trước bạn tự hỏi là tại sao **action** được tạo bởi **createAction** là có 2 lựa chọn để truy xuất **types**, thì mục đích để **reducer** của bạn sẽ ngắn gọn hơn một chút nữa đấy. Dựa vào tính chất **computed property** của **es6** chúng ta hoàn toàn có thể viết lại như sau:

const counter = createReducer({ count: 0 }, {

[increment]: state => ({ count: state.count + 1 }),

[decrement]: state => ({ count: state.count - 1 })

}

Do **increasement** và **descreasement** không phải là một **string**, **computed property** sẽ tự động gọi hàm **toString()**, tương tự như **[increasement.toString()]**.

**Refactor lại toàn bộ store ban đầu**

cùng nhìn lại **store** ban đầu mà chúng ta tạo ra, đã được chuyển sang hoàn toàn với các hàm của **redux-toolkit**:

// store.js

import { configureStore, createAction, createReducer } from '@reduxjs/toolkit';

// actions

const increasement = createAction('increasement')

const decreasement = createAction('decreasement')

const initialState = { count: 0 }

function rootReducer = createReducer(initialState, {

[increasement]: state => ({ count: state.count + 1 })

[decreasement]: state => ({ count: state.count - 1 })

})

const store = configureStore({

reducer: rootReducer

})

export default store

**redux-toolkit** rõ ràng giúp **redux** trở nên rõ ràng, ngắn gọn hơn nhiều, tuy nhiên đòi hỏi bạn sẽ phải hiểu rõ về **redux** trước, sau đó áp dụng **redux-tookit** sẽ thấy khá là sướng , bài viết này mình đã giới thiệu một số **api** chính của **redux-toolkit**, và đó chưa phải là tất cả, **redux-toolkit** vẫn còn nhiều điều hay ho để chúng ta tìm hiểu, mình sẽ giới thiệu ở những phần sau.