**Redux Toolkit**

Reference :

طريقة تطبيق redux خطوة بخطوة

<https://redux-toolkit.js.org/tutorials/quick-start>

**ما هو Redux Toolkit؟**

* مكتبة رسمية من فريق **Redux** لتسهيل كتابة الكود.
* تعتبر **الطريقة الموصى بها رسميًا** لاستخدام Redux.
* تحل مشاكل Redux التقليدي مثل:
  + كثرة الكود (boilerplate code).
  + صعوبة كتابة Reducers و Actions بشكل يدوي.
  + إدارة الـ Store بشكل معقد.

**🟢 أهم المميزات**

1. **تقليل الكود**: بدال ما تكتب Action Types + Action Creators + Reducers منفصلة، بتكتبهم بمكان واحد.
2. **استخدام Immer**: يخليك تعدل على الـ state مباشرة (وكأنه mutable) وهو بيحافظ على الـ immutability.
3. **تجميع المنطق في Slice**: كل جزء من التطبيق (auth, todos, cart…) ممكن يكون له slice خاص.
4. **مبني مع أدوات جاهزة**: مثل Redux DevTools و Thunk Middleware.
5. **تجربة مستخدم أبسط**: في إنشاء store, reducers, async logic.

شو الللي بصير فعليا :

A screenshot of a computer

AI-generated content may be incorrect.

**🔄 تتبع العملية خطوة بخطوة**

**1️⃣ المستخدم يتفاعل مع الـ UI**

مثلاً يضغط زر Add Todo:

<button onClick={() => dispatch(addTodo("Learn Redux Toolkit"))}>

Add Todo

</button>

**2️⃣ استدعاء الـ Action**

الدالة addTodo("Learn Redux Toolkit") بترجع **action object**:

{

type: "todos/addTodo",

payload: "Learn Redux Toolkit"

}

لاحظي إن **type** فيه اسم الـ slice ("todos") + اسم الـ reducer ("addTodo").  
وهذا بيساعد store يميز لأي reducer يروح.

**3️⃣ إرسال الـ Action للـ Store**

dispatch بيرسل الـ action إلى **store**.

**4️⃣ الـ Store يوزّع الـ Action على كل Reducers**

* الـ store ما بيختار reducer واحد فقط.
* بدل ذلك، بيرسل الـ action لكل reducers موجودين في الـ store.
* في حالتنا:
  + counterReducer بيشوف type: "todos/addTodo" → مش تابعه → يرجع نفس state تبع counter.
  + todosReducer بيشوف type: "todos/addTodo" → آه هذا إلي! → يعدل حالته ويرجع نسخة جديدة.

**5️⃣ تحديث الـ State Tree**

الـ store بيجمع النتائج من كل reducers وبيعمل state جديد:

{

counter: { value: 0 },

todos: ["Learn Redux Toolkit"]

}

**6️⃣ إشعار React بوجود State جديد**

* أي كومبوننت عامل useSelector بيرجع يقرأ الـ state الجديد.
* React بتعمل **re-render** وتعرض النتيجة الجديدة على الشاشة.

**✨ وظيفة الـ Store**

الـ **store** هو "العقل المركزي" في Redux، ومهمته الأساسية 3:

1. **تخزين state التطبيق كله** في مكان واحد (state tree كامل).  
   مثال:
2. {
3. counter: { value: 3 },
4. todos: ["task1", "task2"]
5. }
6. **استقبال الـ actions** (عن طريق dispatch) وتمريرها لكل reducers.
7. **إشعار الـ UI** بأي تغييرات صارت في الـ state → حتى React تعيد render.
8. **Store contains 2 part**

**1-state object wich contains all the app states we can use this state by calling custom hook named useSelector()**

**2- reducer object wich contain all app reducers and we can call it by using dispatch() function wich take the action object {type:action name,payload:{}} and the payload if exist .**

**📊 ملخص التدفق**

User clicks button

↓

dispatch(action)

↓

store يستقبل action

↓

store يمرر action لكل reducers

↓

reducer المطابق يعدل حالته

↓

store يجمع كل النتائج في state جديد

↓

React تعمل re-render

↓

UI يتحدث

Is slice must handle only one state ?

الـ **slice** في Redux Toolkit مش لازم يضل مربوط بحالة (state) وحدة فقط، هو يمثل **جزء (slice) من الـ store**.

**📌 كيف يعني؟**

* عادةً من الأفضل إن كل slice يمثّل **موضوع محدد** (domain) في التطبيق.
* جواته ممكن يكون عندك **أكثر من property** في الـ state، مش بس وحدة.

**✅ مثال 1: Slice يحتوي state واحد (Counter)**

const counterSlice = createSlice({

name: "counter",

initialState: { count: 0 },

reducers: {

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

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

}

})

هنا عندك فقط count.

**✅ مثال 2: Slice يحتوي عدة قيم (User)**

const userSlice = createSlice({

name: "user",

initialState: {

name: "",

email: "",

isLoggedIn: false

},

reducers: {

login: (state, action) => {

state.name = action.payload.name;

state.email = action.payload.email;

state.isLoggedIn = true;

},

logout: (state) => {

state.name = "";

state.email = "";

state.isLoggedIn = false;

}

}

})

هنا الـ slice يتعامل مع **state أكثر من واحد**: name, email, isLoggedIn.

**✅ مثال 3: Slice يحتوي object كبير (Cart)**

const cartSlice = createSlice({

name: "cart",

initialState: {

items: [],

total: 0,

isOpen: false

},

reducers: {

addItem: (state, action) => {

state.items.push(action.payload);

state.total += action.payload.price;

},

toggleCart: (state) => {

state.isOpen = !state.isOpen;

}

}

})

هنا عندك state متشعب: items, total, isOpen.

**✨ الخلاصة**

* الـ slice مش شرط يحتوي **state واحد فقط**.
* الأفضل كل slice يعالج **مجال (feature/domain) واحد**.
* جوا هذا المجال ممكن يكون عندك state كثيرة.

**Redux DevTools**

هي **إضافة للمتصفح** (Browser Extension) تساعد المطورين على تتبع حالة الـ Redux Store (الـ States والـ Actions) وتسهّل عملية الـ Debugging.

**✅ المميزات (Advantages):**

1. **عرض جميع الـ Actions**
   * تقدر تشوف كل الأكشنات اللي صارت في الـ Store بالترتيب.
2. **إظهار حالة الـ State قبل وبعد كل Action**
   * بيبينلك الـ State القديم والـ State الجديد بعد تنفيذ الأكشن.
3. **إمكانية التراجع أو التقدم بين الأكشنات (Time Travel Debugging)**
   * عن طريق الـ Slider الموجود أسفل الأداة تقدر ترجع خطوة لورا أو لقدّام لتشوف كيف تغيّر الـ State مع كل أكشن.

A screenshot of a computer

AI-generated content may be incorrect.

**The Problem of Async Logic in Redux**

**Redux store doesn't know anything about async logic. It only knows how to synchronously dispatch actions, update the state by calling the root reducer function, and notify the UI that something has changed. Any asynchronicity has to happen outside the store.**

**Store in redux**  مهئ انو يتعمل فقط مع sync logic يعني لو حاولت اغير احد ال

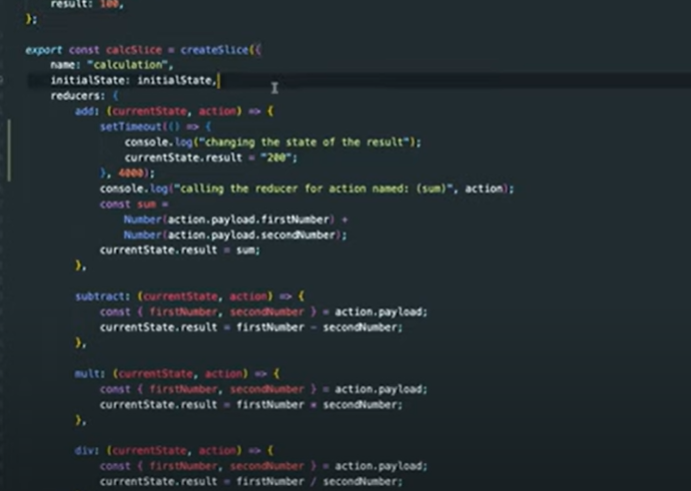
state المخزنة في ال store داخل async function as fuction contains api request حيعطيني ارور ليش لانو زي ما حكينا الستور مهئ بالوضع الطبيعي للتعامل مع sync فلما بزبط اغير ال state من داخل كود asyncز

لكن احنا بنحتاج مثلا في داخل async function contains api request or read from file …

انو نغير قيمة ال state شو الحل ؟

لازم نضيف شوية إضافات لل redux ليعرف يتعامل مع async

Ex





هاد السطر حيضرب ارور والسبب انو بنحاول نعدل على الحالة من داخل فنكشن غير متزامن

**🔹 الحل:**

نقوم بإضافة **طبقة جديدة** اسمها **Middleware**.  
وظيفتها: استقبال **Thunk Function** يتم إرساله من الـ Component عند تنفيذ dispatch.

* هذا الـ **middleware** هو المسؤول عن تشغيل الكود غير المتزامن (Asynchronous Code) مثل **API Request**.
* بعد انتهاء تنفيذ الكود ورجوع **الـ Response** مع البيانات المطلوبة:
  + يقوم الـ middleware بعمل **dispatch جديد** إلى الـ Store.
  + هذا الـ dispatch يحتوي على **Action Object** فيه:
    - type (نوع الأكشن)
    - payload (البيانات القادمة من الـ response).

بعدها:

* الـ Reducer يستقبل هذا الـ action ويكمل شغله بشكل طبيعي يحدث الـ state

**✅ الفكرة الأساسية**

بما إن الـ **Store** غير قادر على تنفيذ كود غير متزامن داخل الـ Reducer مباشرة:

* نتحايل على هذا الموضوع بإضافة **Middleware**.
* الـ Middleware ينفذ الكود غير المتزامن، ثم يرسل النتيجة كـ **Dispatch عادي** إلى الـ .Store.

A screenshot of a computer

AI-generated content may be incorrect.

A diagram of a software system

AI-generated content may be incorrect.

1-create store file -> crete store using configerStore() function which import from redux toolkit حيكون ال reducer في البداية فاضي

A screenshot of a computer

AI-generated content may be incorrect.

2- create slice named اسم featuresSlice file