وب سایت :

https://redux-toolkit.js.org/

**Redux Toolkit (RTK)**

Redux Toolkit یا RTK یک لایبرری رسمی از تیم Redux است که هدفش **سادگی و استانداردسازی کار با Redux** است. به‌طور خلاصه، RTK مشکلات و boilerplate های سنتی Redux را حل کرده و توسعه‌دهنده را قادر می‌سازد سریع‌تر، امن‌تر و مقیاس‌پذیرتر state مدیریت کند.

**چرا Redux Toolkit استفاده می‌کنیم ؟**

دلایل اصلی استفاده از RTK:

1. **کاهش boilerplate**
   * در Redux سنتی شما باید actions, action types, reducers و store را جداگانه تعریف کنید.
   * RTK با createSlice و configureStore این روند را ساده می‌کند.
2. **State امن و بدون mutation**
   * RTK از Immer داخلی استفاده می‌کند تا بتوانید state را به‌صورت mutable بنویسید، ولی واقعاً immutable شود.
3. **سهولت در مدیریت Async actions**
   * با createAsyncThunk می‌توان API callها را ساده و قابل مدیریت کرد، همراه با lifecycle states: pending, fulfilled, rejected.
4. **بهترین روش‌های استاندارد**
   * RTK راهنما و استانداردهای مشخصی دارد که از بروز خطاهای رایج جلوگیری می‌کند.

**تفاوت RTK با Context و دیگر state manager ها**

****

نکته: Context

خوب است برای state محلی و کوچک مثلاً theme یا auth state ساده ، اما برای state بزرگ،

RTK مناسب‌تر و امن‌تر است.

## برای چه پروژه‌هایی مناسب است؟

* پروژه‌های **متوسط تا بزرگ** با state global پیچیده
* پروژه‌هایی که نیاز به **DevTools و debugging** دارند
* پروژه‌هایی که API-heavy هستند و نیاز به **Async logic مدیریت شده** دارند
* تیم‌های بزرگ که نیاز به **استاندارد سازی کد** دارند

❌ پروژه‌های خیلی کوچک یا state ساده ممکن است RTK اضافه بار باشد.

**توضیح store :**

Store یعنی **مغز مرکزی مدیریت داده‌ها در اپلیکیشن**.  
همه‌ی stateهای مهم مثل todos, user, cart و … داخل Store نگهداری می‌شن.

در واقع، store یه **آبجکت جاوااسکریپتی** هست که:

* کل state فعلی برنامه رو نگه می‌داره
* به reducerها گوش می‌ده
* و وقتی state عوض می‌شه، componentهایی که به اون وصل شدن از طریق useSelector رو **به‌روزرسانی می‌کنه**.

**اطلاعات کجا ذخیره میشه ؟**

Redux Store فقط داخل **حافظه‌ی (RAM) مرورگر در زمان اجرا** نگهداری میشه.  
یعنی:

* داخل هیچ فایل یا فولدری ذخیره نمی‌شه
* داخل دیتابیس یا localStorage هم به صورت پیش‌فرض نمی‌ره
* فقط وقتی صفحه بازه، توی حافظه (memory) نگهداری میشه

**پس وقتی صفحه رفرش میشه چی میشه ؟**

وقتی صفحه **رفرش بشه یا بسته بشه**:

* Store از بین می‌ره
* چون فقط در حافظه‌ی موقتی مرورگر نگهداری میشه

ولی اگر بخوای state رو **ذخیره‌ی دائم** کنی، باید از کتابخونه‌هایی مثل:

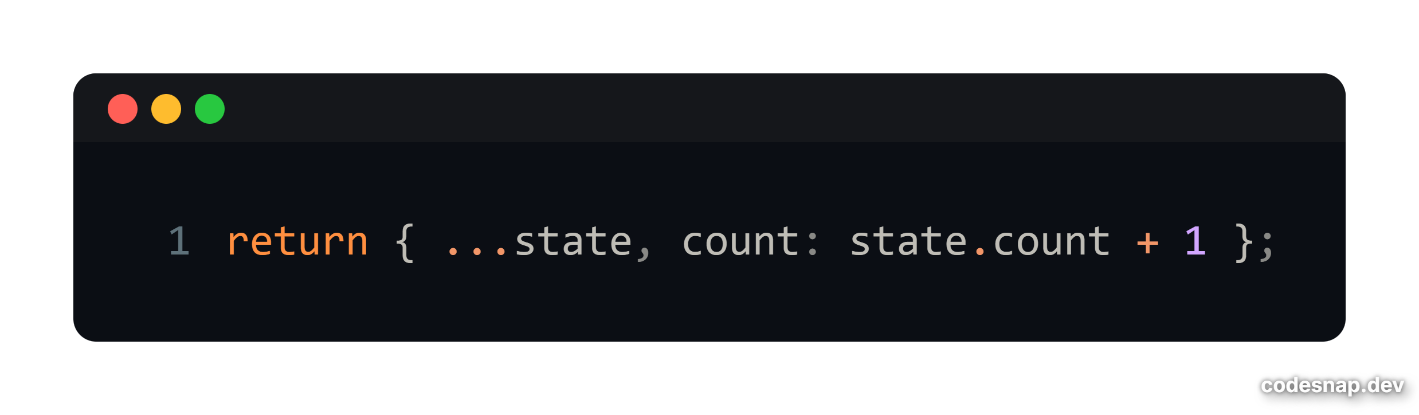
* redux-persist استفاده کنی   
  که Store رو داخل **localStorage**  ذخیره می‌کنه  
  و وقتی کاربر برمی‌گرده، دوباره اون state رو بارگذاری می‌کنه.

## خلاصه store

Store مثل مغز برنامه‌ست که همه‌ی stateها داخلش نگهداری می‌شن.  
خودش فقط یه آبجکت در حافظه است (نه فایل واقعی).  
با رفرش صفحه از بین می‌ره.  
برای ذخیره‌ی دائمی باید از کتابخونه‌هایی مثل redux-persist استفاده کنیم.  
کل برنامه از طریق Provider به Store وصل میشه.

**توضیح immer / immutable**

در Redux معمولی، ما نمی‌تونستیم state رو مستقیماً تغییر بدیم،  
چون Redux باید بتونه تشخیص بده چه زمانی state عوض شده.  
برای همین باید همیشه یه **کپی جدید از state** می‌ساختیم  
با استفاده از {...state} یا map, filter و ..



در **Redux Toolkit**، یه کتابخونه داخلی به نام **Immer** وجود داره   
که این کار رو **به صورت خودکار پشت صحنه انجام می‌ده**.

بنابراین ما می‌تونیم خیلی راحت‌تر بنویسیم:



در ظاهر به نظر می‌رسه داریم مستقیماً state رو تغییر می‌دیم،  
ولی در واقعیت، **Immer یه کپی جدید از state می‌سازه**  
و تغییرات رو روی اون اعمال می‌کنه.

در Redux معمولی باید خودمون کپی جدید از state می‌ساختیم،  
ولی در Redux Toolkit به کمک کتابخونه‌ی **Immer**،  
این کار به‌صورت خودکار انجام می‌شه.  
پس ما می‌تونیم state رو مثل حالت عادی تغییر بدیم،  
ولی در پشت صحنه هنوز immutable باقی می‌مونه.»

**نصب و راه اندازی :**

npm install @reduxjs/toolkit react-redux

**ساخت Store در پروژه :**



**ساخت slice :**



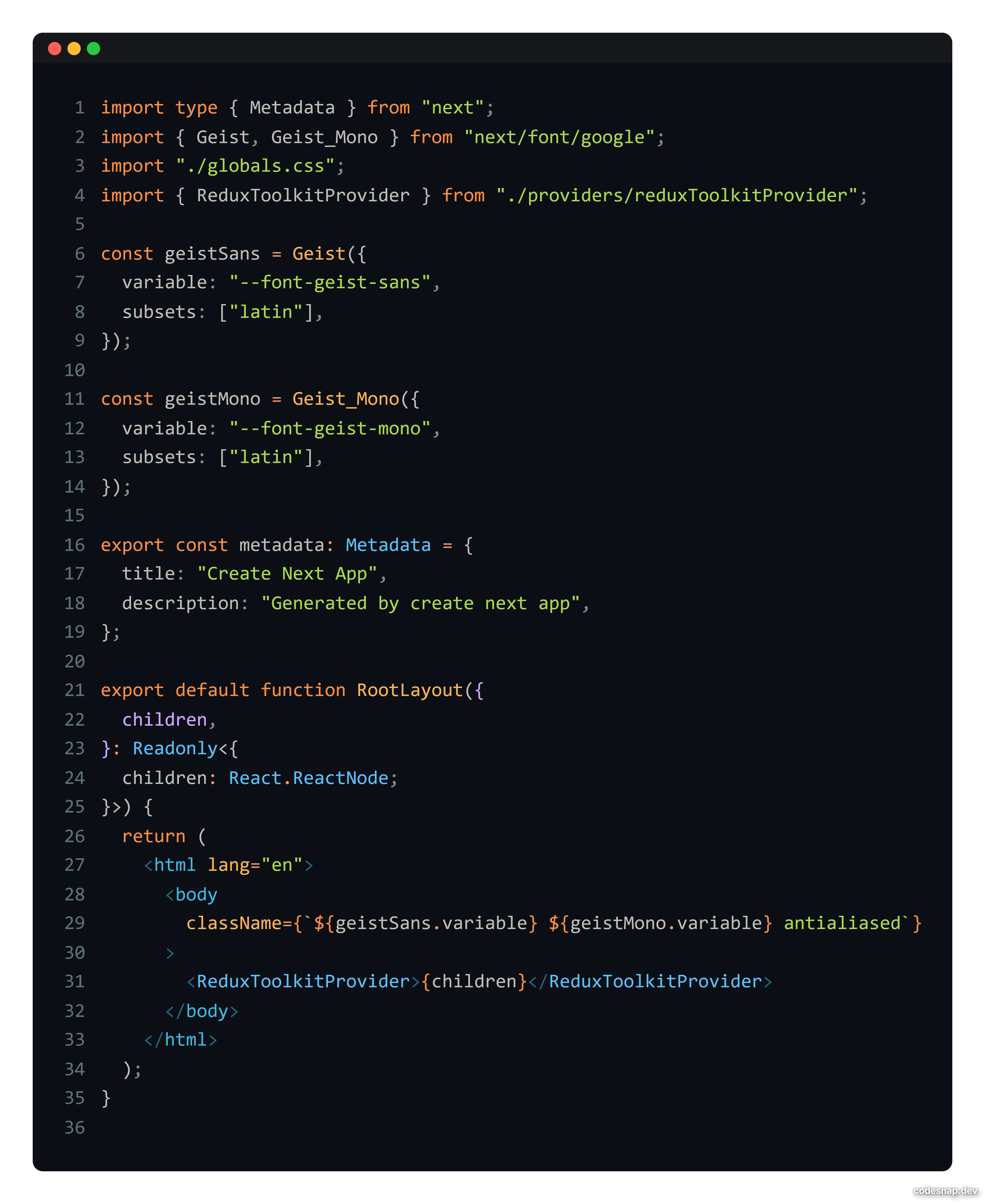
**نحوه استفاده :**



**ایجاد کردن provider :**



**استفاده از provider در layout.tsx :**



**مدیریت Async Actions با createAsyncThunk ( API و هر انچه که نیاز به Async داره )**



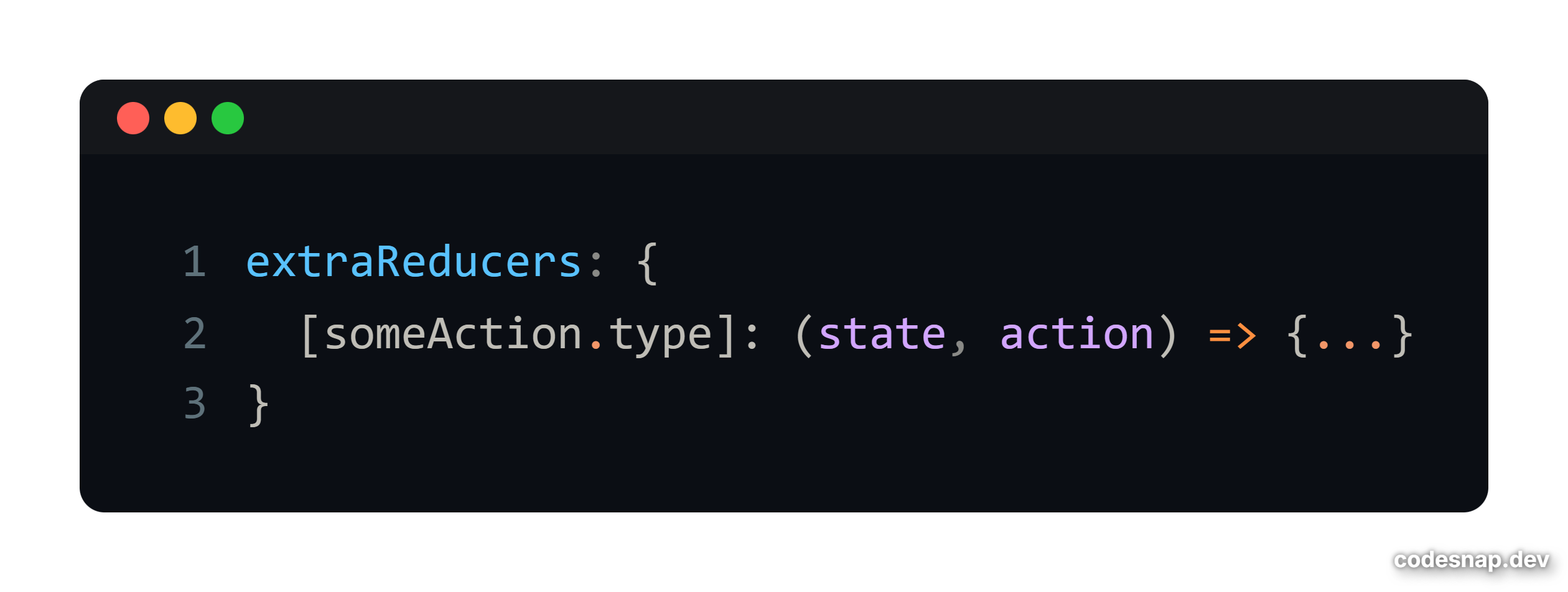
# extraReducers

در Redux Toolkit، createSlice دو نوع reducer دارد:

1. **reducers معمولی** (reducers)
   * برای action های داخلی slice استفاده می‌شود.
   * خود RTK برای شما action creator می‌سازد.
   * مثال: increment, decrement
2. **extraReducers**
   * برای مدیریت **actionهایی که در خارج از slice ساخته می ‌شوند**.
   * معمولاً برای **AsyncThunkها** یا actionهای دیگر slice استفاده می‌شود.
   * در واقع اینجا می‌گویید: "وقتی این action اتفاق افتاد، state را چطور تغییر دهم".

# builderچیست؟

* builder **یک شی object-oriented** است که Redux Toolkit برای extraReducers فراهم می‌کند.
* مزیت آن نسبت به **object notation**:



**در صورت نبودنش به این شکل باید استفاده میشد که استفاده ازش کار ما رو راحت می کنه .**

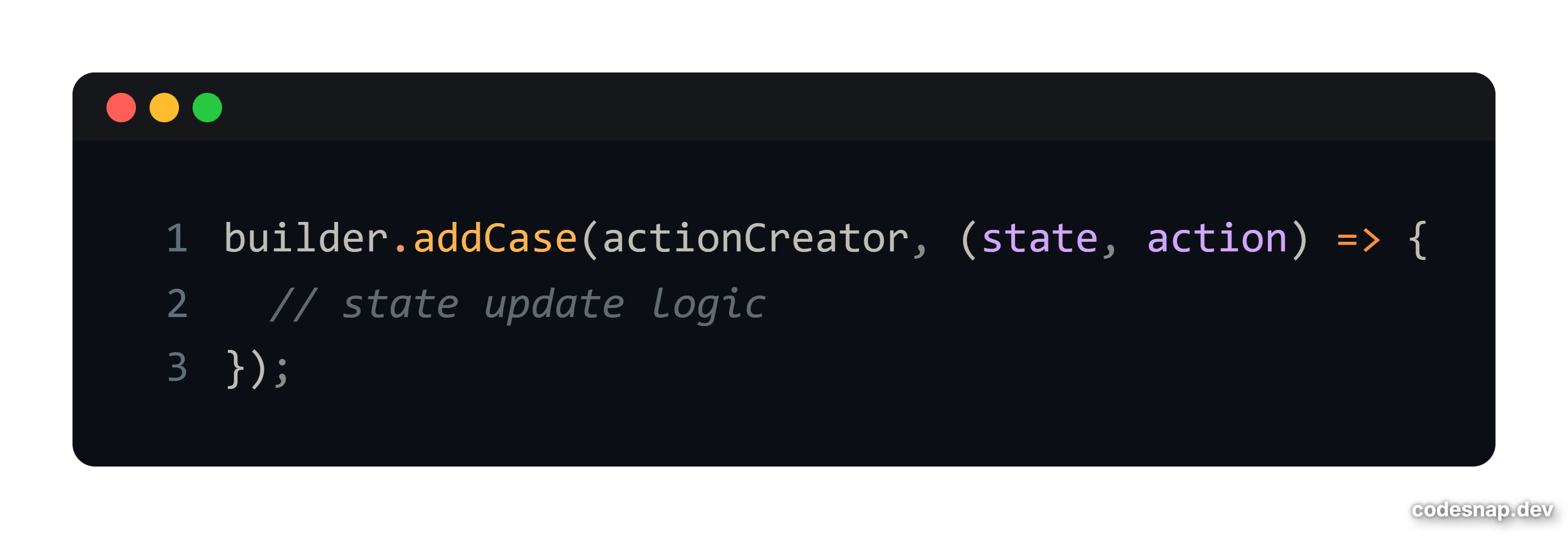
Builder **type-safe** است و TypeScript را بهتر پشتیبانی می‌کند.

مخصوصاً برای **asyncThunk ها** مناسب است، چون TS می‌تواند نوع

action.payload را تشخیص دهد.

# builder.addCase

* متد addCase روی builder برای **اضافه کردن handler به یک action خاص** استفاده می‌شود.
* سینتکس:



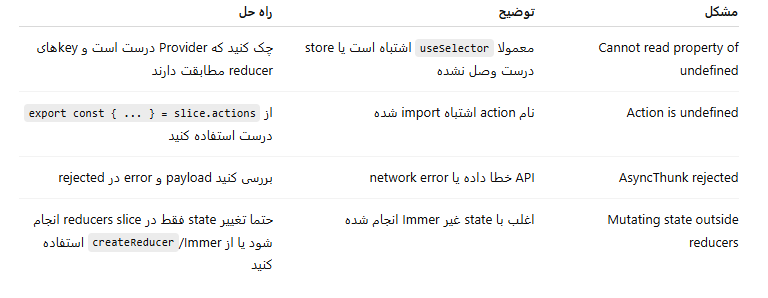
اگر action یک createAsyncThunk باشد، می‌توانید سه حالت آن را اضافه کنید:

1. **pending** → درخواست شروع شده
2. **fulfilled** → درخواست موفق
3. **rejected** → درخواست با خطا

# نکات حرفه‌ای best practice

1. **استفاده از builder توصیه می‌شود**
   * type-safe است و برای AsyncThunkها کامل است.
2. **برای AsyncThunk ها همیشه ۳ حالت را مدیریت کنید**
   * pending لودینگ
   * fulfilled ذخیره داده
   * rejected ذخیره خطا
3. **state mutation با Immer**
   * داخل addCase می‌توانید مستقیماً state.data = action.payload بنویسید و RTK خودش immutable handling را انجام می‌دهد.
4. **می‌توان چند addCase پشت سر هم نوشت**
   * به جای یک object با کلید action type، builder chain خیلی خواناتر و مقیاس‌پذیرتر است.

**مشکلات رایج و راه حل ‌ها**

****

## خلاصه Flow RTK

Slice → Actions + Reducers

Store → ConfigureStore با slices

Component → useSelector + useDispatch

Async → createAsyncThunk + extraReducers

**ساختار فولدر :**

**با RTK Query**



**بدون RTK Query**

