28.Redux Toolkit (RTK) 相比传统 Redux 解决了哪些痛点?

Redux Toolkit (RTK) 面试题

Q1: 什么是 Redux Toolkit (RTK)，它被设计用来解决什么核心问题?
A1: Redux Toolkit (RTK) 是 Redux 官方推荐的、用于高效进行 Redux 开发的工具集。它旨在解决传统 Redux 开发中的几个核心痛点，其主要目标包括：

- **简化 Store 配置**：自动化和简化 Redux Store 的创建过程。
- **减少模板代码**：消除手动编写大量 Action Types、Action Creators 和 switch 语句的冗余工作。
- **简化不可变更新**：通过内置 Immer.js，让状态的不可变更新变得更简单直观。
- **内置最佳实践**：开箱即用地集成了常用的中间件（如 Redux Thunk）和 Redux DevTools 的配置。

Q2: 相比传统 Redux，使用 RTK 的 `configureStore` API 有什么优势?
A2: `configureStore` 极大地简化了传统 Redux 中繁琐的 Store 配置流程，其主要优势体现在：

- **自动合并 Reducers**：你只需提供一个包含各个 slice reducer 的 `reducer` 对象，`configureStore` 会自动为你调用 `combineReducers`。
- **内置常用中间件**：它默认集成了 `redux-thunk` 中间件，无需手动配置即可处理异步 Action。同时它还会自动添加用于开发环境检查的中间件，如检查状态是否意外突变。
- **自动配置 DevTools**：它会自动启用并配置 Redux DevTools Extension，让你开箱即用地享受强大的调试功能。
- **代码简洁**：将原本需要多步（合并 reducer、创建 store、应用中间件、组合 enhancers）的操作，简化为一次函数调用。

Q3: 请解释一下 RTK 中的 `createSlice` 是什么，以及它是如何帮助我们减少模板代码的?
A3: `createSlice` 是 RTK 的核心 API 之一，它接收一个切片（slice）的名称、初始状态（initialState）和一个包含 reducer 函数的对象，并自动生成对应的 action creators 和 action types。它通过以下方式减少模板代码：

- **自动生成 Action Types**：`createSlice` 会根据你提供的 `name` 和 `reducers` 对象中每个函数的键名，自动生成 action type 字符串（例如：`'counter/increment'`）。你不再需要手动定义这些常量。
- **自动生成 Action Creators**：对于 `reducers` 对象中的每一个函数，`createSlice` 都会自动创建一个同名的 action creator 函数。例如，一个名为 `increment` 的 reducer 会对应生成一个 `increment()` action creator。

- **替代 `switch` 语句**：`reducers` 对象本身就是一个查找表，将 action type 直接映射到对应的更新函数，从而完全消除了在传统 reducer 中冗长的 `switch` 语句。

Q4: 在 `createSlice` 的 reducers 中，我们可以编写像 `state.value += 1;` 这样“看似可变”的代码，这是为什么？这样做有什么好处?
A4: 这是因为 `createSlice` 内部集成了 **Immer.js** 库。

- **工作原理**：当你编写看似直接修改 `state` 的代码时，Immer 会在底层进行追踪。它会创建一个临时的草稿状态（draft state），并将你的修改应用到这个草稿上。当你的 reducer 函数执行完毕后，Immer 会根据草稿的变化安全地生成一个全新的、不可变的最终状态，而原始 `state` 保持不变。
- **好处**：
  - **提升开发体验**：代码写法更直观、更简洁，就像操作普通 JavaScript 对象一样，降低了心智负担。
  - **减少错误**：避免了手动进行对象扩展（`...spread`）或 `Object.assign` 时可能出现的遗漏或错误，从而降低了意外直接修改原始状态的风险。
  - **保证不可变性**：在享受便捷写法的同时，仍然严格遵守了 Redux 的核心原则——状态不可变性。

Q5: RTK 是如何处理异步操作的？请简述 `createAsyncThunk` 的作用。
A5: RTK 推荐使用 `createAsyncThunk` 来标准化异步操作流程。

- **作用**：`createAsyncThunk` 是一个接收 action type 字符串和一个返回 promise 的 "payload creator" 函数的工具。它会为你自动 dispatch 三种不同生命周期的 action：
  - `pending`：在异步操作开始时触发。
  - `fulfilled`：在 promise 成功解决时触发，其 payload 为 promise 的返回值。
  - `rejected`：在 promise 被拒绝时触发，其 payload 包含错误信息。
- **使用方式**：你可以在 `createSlice` 的 `extraReducers` 字段中监听并处理由 `createAsyncThunk` 生成的这些 action，从而分别更新加载中、成功或失败时的状态。这使得异步逻辑的管理更加结构化和一致。

Q6: 在面试中，如果被问到“为什么选择 Redux Toolkit？”，你会如何有条理地回答?
A6: 我会从以下几个方面来阐述选择 Redux Toolkit 的理由：

- **第一，显著减少模板代码**：RTK 的 `createSlice` API 可以根据 reducer 函数自动生成 action types 和 action creators，并用一个简单的对象取代了冗长的 `switch` 语句，极大地提升了代码的简洁性。
- **第二，极大地简化了 Store 配置**：通过 `configureStore`，我们可以用一行代码完成 store 的创建，它自动集成了 Redux Thunk 用于异步处理，并配置好了 Redux DevTools，真正做到了开箱即用。
- **第三，提升了开发体验和代码健壮性**：RTK 内置了 Immer.js，允许我们在 reducer 中用直观的方式“直接修改”状态，而 Immer 会在底层确保状态的不可变性。这不仅让代码更易读，也有效防止了意外修改状态导致的 bug。

- **第四，标准化了异步请求流程**：`createAsyncThunk` 为 API 调用等异步操作提供了一套标准的模式，自动管理 `pending`，`fulfilled`，`rejected` 这三种状态，使异步代码更加一致且易于维护。
- **总结**：总而言之，Redux Toolkit 作为 Redux 官方推荐的工具集，通过解决传统 Redux 的核心痛点，使得代码更简洁、开发效率更高、维护性更强，是目前任何新老项目中使用 Redux 的最佳实践。