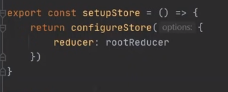
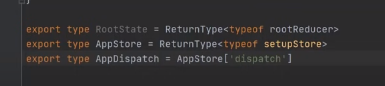
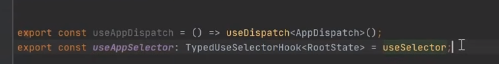
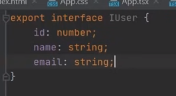
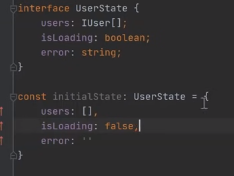
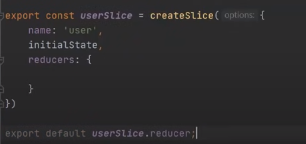
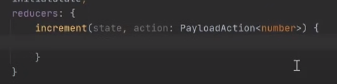
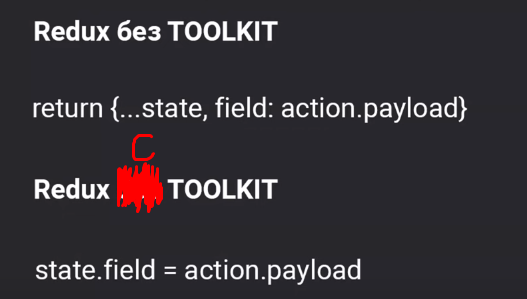
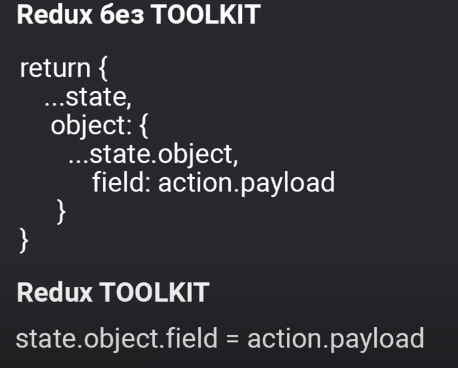
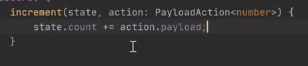
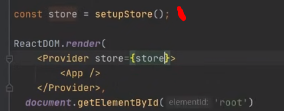
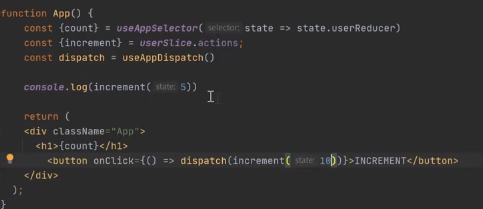
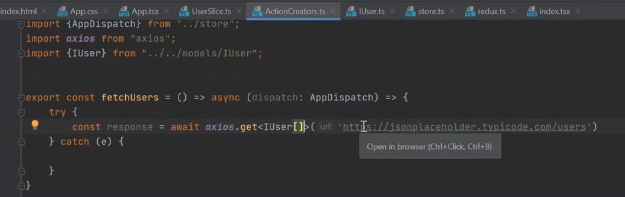
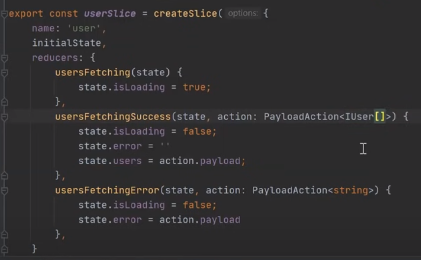
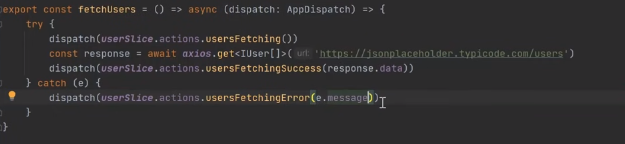
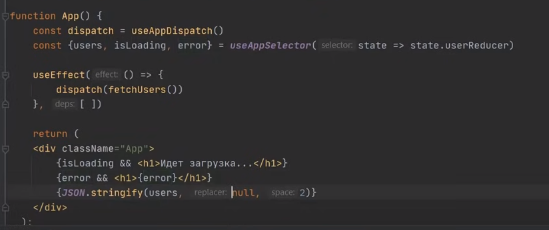
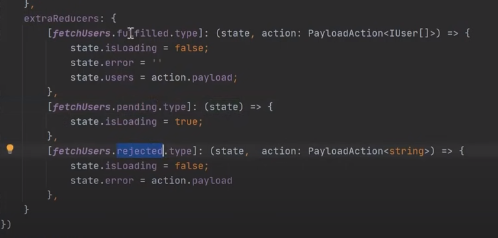
1) Установка.  
  
Обычный create react app typescript.  
.  
  
2) Начало разработки. Конфигурация store.  
  
Создаём rootReducer с помощью combineReducers.  
  
Создаём setupStore с помощью configureStore.  
Стоит отметить, что когда мы используем reduxToolKit, то нам не нужно подключать инструменты разработчика для отладки и redux thunk middleware, поскольку всё уже идет из коробки.  
  
  
  
Теперь получаем типы state, store, dispatch.  
  
  
**3) Кастомные хуки для работы с redux.**  
  
-> useAppDispatch – обычный диспатч, но с типизацией.  
-> useAppSelector – обычный селектор, но с типизацией (с типом state).  
  
  
**4) Redux toolkit slice.**  
В редакс reducer – чистая функция.  
В редакс toolkit есть slice, это как некоторая обёртка над reducer.

Создаем папку models, объявим файл IUser.ts, типизируем пользователя, поскольку будем работать со списком пользователей.  
  
  
  
В папке reducers, создаём файл UserSlice.ts, в нём и будет наш слайс.  
Пишем интерфейс для состояния и начальное состояние.  
  
  
  
Теперь создадим сам слайс с помощью createSlice, принимает объект с опциями(name, initialstate, reducers(поле аналогично switch case, которое используем в обычном reducer)).  
  
  
  
Вытащим отдельно Reducer и перенесём его в rootReducer.  
  
Добавим к нашему пользователю счётчик.  
count в состоянии.

Как уже было написано в reducers у слайса мы делаем то же самое, что и в switch case.  
  
Мы также можем типизировать Payload.  
  
  
Теперь рассмотрим такой пример.  
  
  
  
  
Получается, нам не обязательно полностью возвращать новый State, используя redux toolkit.

  
  
Теперь можно использовать и изменять Store в каком либо компоненте.  
Оборачиваем наше приложение в Provider.  
  
  
  
Как уже писалось, userSlice содержит actions и reducer.  
Мы не должны создавать в ручную ни типы для Action Creators, ни их самих.  
  
  
  
Это всё был вводный пример, чтобы просто познакомиться с rtk.  
Убираем count.  
  
**5) Обработка стандартного сценария.**Получение данных, обработка ошибки и загрузки.  
В папке reducers создадим actionCreators.  
  
Установим Axios.  
  
  
Теперь нам необходимо указать Reducer.  
  
  
  
  
  
Реализуем этот функционал в приложении.  
  
  
Redux Toolkit позволяет упростить обработку таких сценариев.  
  
  
Но в таком случае мы ничего не деспатчим, как redux узнает об этом?  
  
Для такого случая в CreateSlice есть поле extraReducers.  
  
А поле reducers можно очистить.  
  
Но обработка ошибки не будет происходить, исправим это, добавив try catch и вызов rejected в catch поле.  
