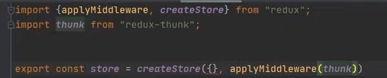
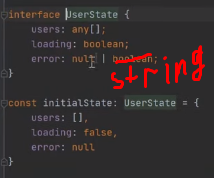
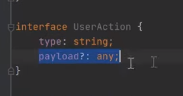
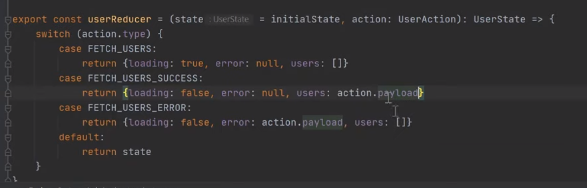
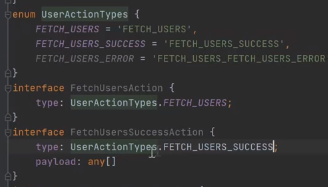
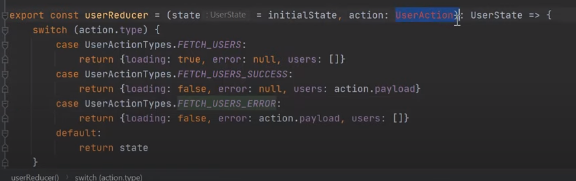
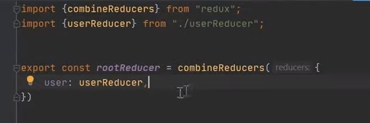
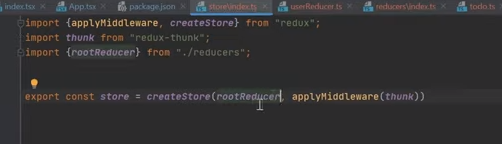
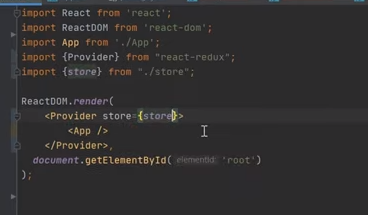
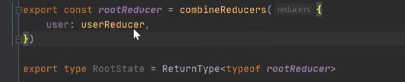
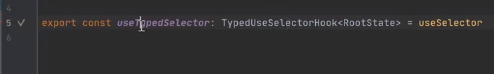
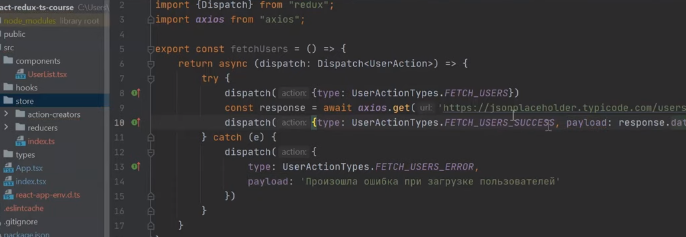
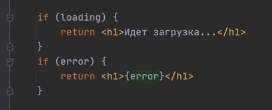
**React & Redux & TypeScript ПОЛНЫЙ КУРС 2021**1) Подготовка. Типизация. Первый Редьюсер.  
  
CRA typescript template.  
  
  
  
Папки types, store, components.  
В store создаем index.ts.  
  
  
  
Создаем папку reducers в папке store и туда кладём userReducer.  
  
В этом файле описываем интерфейсы для state, action.  
Создаём так же начальный state.  
И саму функцию userReducer.   
  
  
  
  
  
  
Но теперь получается, что в payload может попасть и ошибка и строка и вообще всё, что угодно.  
  
Предотвратим это, создав интерфейсы к каждому Action.  
  
  
  
  
  
  
Теперь нужно разделить типы и логику, в папку types выносим все интерфейсы и enum.  
**2) CombineReducer, Provider.**  
В папке reducers создадим index.ts, в котором создадим combineReducer, который будет объединять все редьюсеры.  
  
  
  
Теперь поправляем store.  
  
  
  
Теперь остается прокинуть состояние в реакт компоненты.  
Делается с помощью провайдера, который находится в реактредакс.  
Передаем туда так же как пропы store.  
  
  
  
Создаём userList.  
Данные из состояния можно выцеплять с помощью хука useSelector.  
  
  
Дефолтный useSelector с типами не дружит.  
  
**3) Создадим свой useTypedSelector.**Сначала type RootState.  
  
  
  
Теперь папку hooks и в нем useTypedSelector.  
  
  
  
  
**4) Action creator.**  
  
Создадим action creators.  
   
  
Воспользуемся этой функцией в useEffect.  
  
Создадим dispatch с помощью useDispatch.  
  
  
Сделаем обычные отловы загрузки и ошибки.  
  
  
**5) Запись с dispatch и actionCreators можно упростить.**  
  
Создадим ещё один хук. useAction.  
Вся суть хука – связь actionCreator с dispatch.  
Функция bindActionCrators.  
  
