React: Redux

AIT
Hi-tech team

НАШИ ПРАВИЛА

Включенная камера

Вопросы по поднятой руке

Не перебиваем друг друга

Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору

Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

Повторим;)

Для чего используется хук useContext?

Что возвращает вызов хука useState?

Как выполнить действие при монтировании компонента?

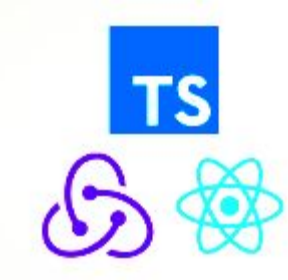
ЦЕЛЬ

Настроить проект и изучить основную концепцию redux

ПЛАН ЗАНЯТИЯ

1. Настройка проекта
2. Введение в redux

Создание проекта React Redux



Установка

Рекомендуемый способ запуска новых приложений с помощью React и Redux Toolkit — использование официального шаблона

Redux Toolkit + TS для Vite

Ссылка на репозиторий:

https://github.com/reduxjs/redux-templates/tree/master/packages/vite-template-redux

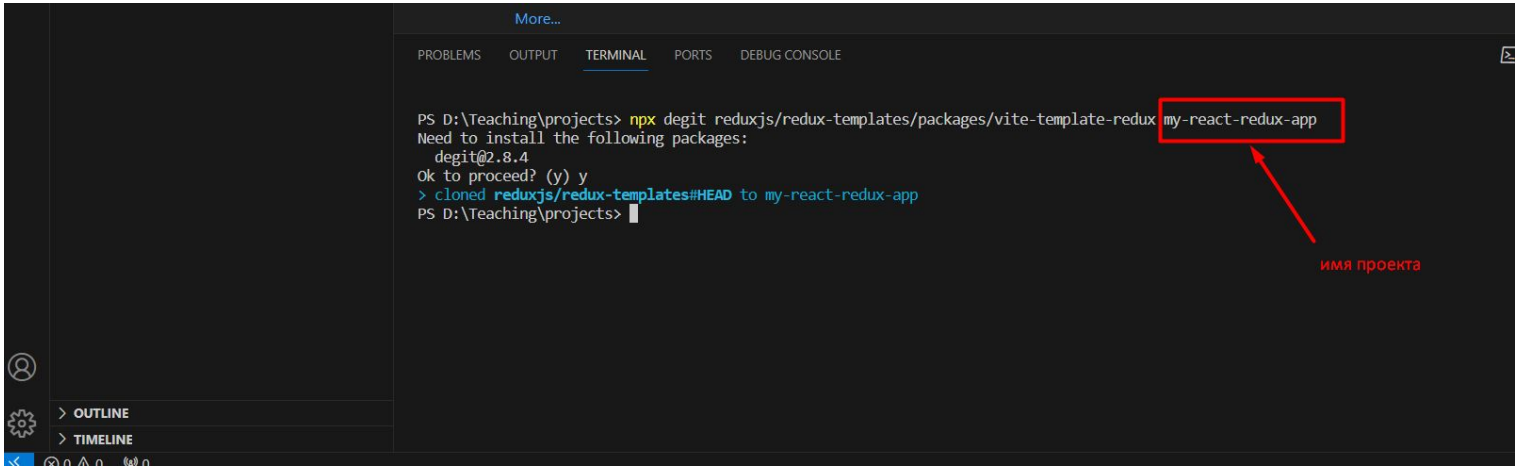
Примечание: Vite — это инструмент сборки, цель которого — обеспечить более быструю и экономичную разработку современных веб-проектов.

Установка

Шаг 1

Откройте VSCode, перейдите в папку, в которой будет лежать ваш проект. В терминале введите следующую команду

```
npx degit reduxjs/redux-templates/packages/vite-template-redux my-app
```



Примечание: у вас на компьютере уже должен быть установлен Node.js

Установка

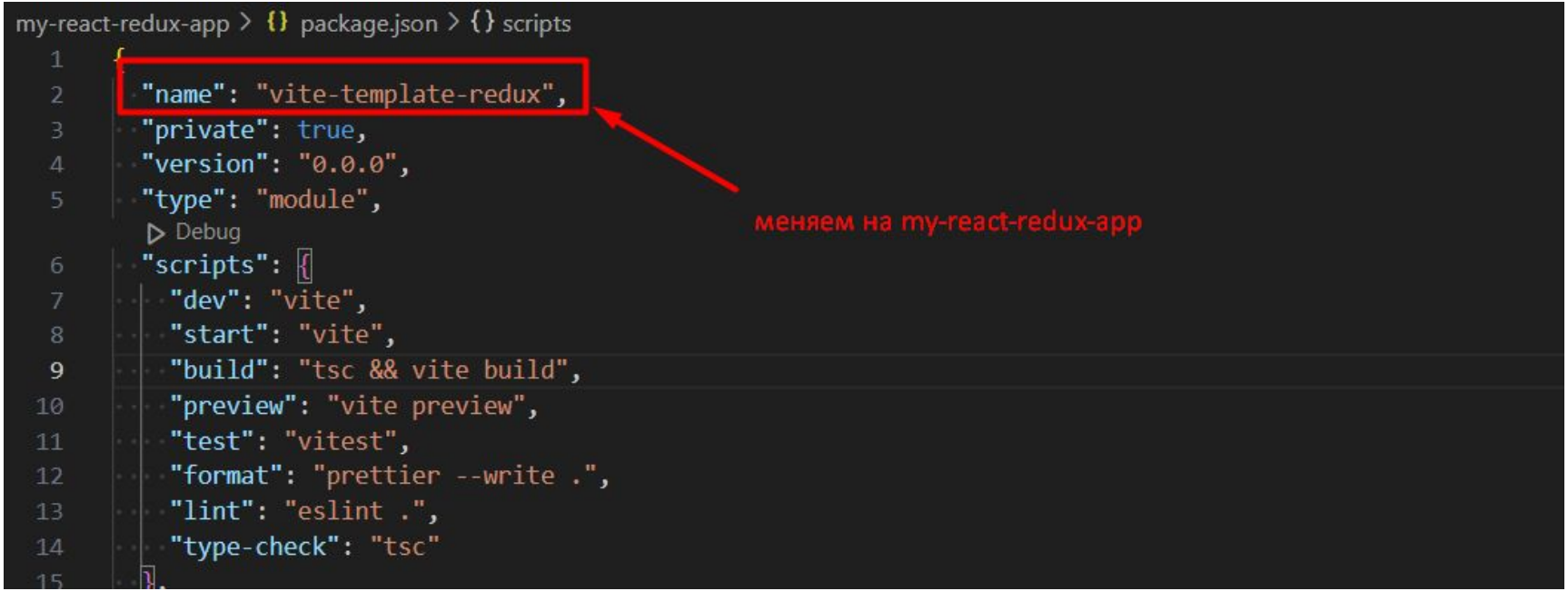
Шаг 2

В VSCode перейдите в папку вашего проекта

Установка

Шаг 3

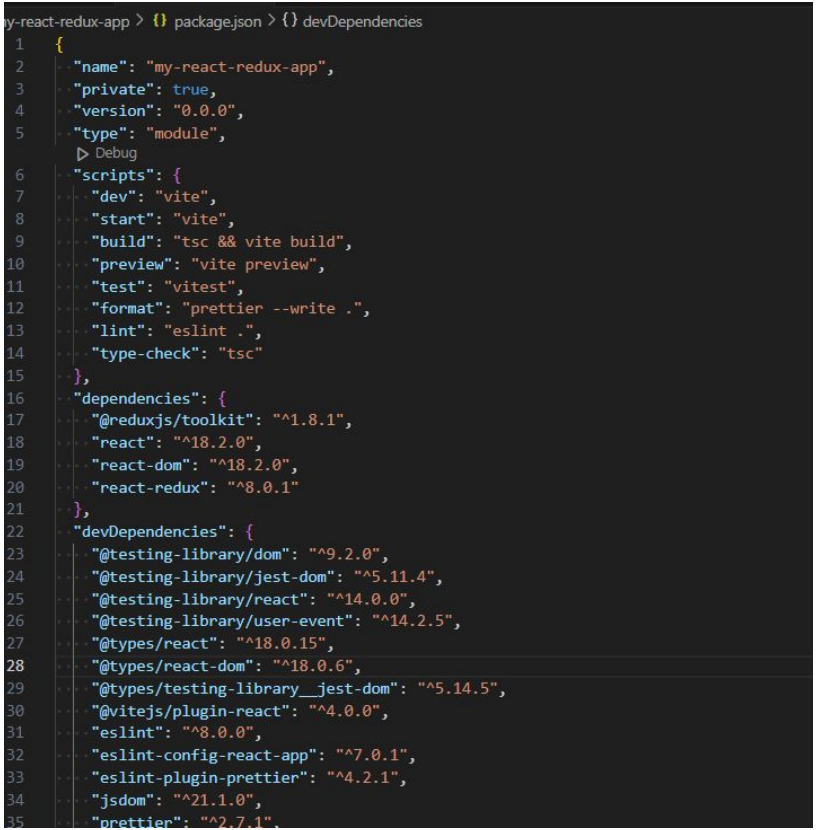
Открываем файл package.json и меняем название нашего проекта



Установка

Шаг 4

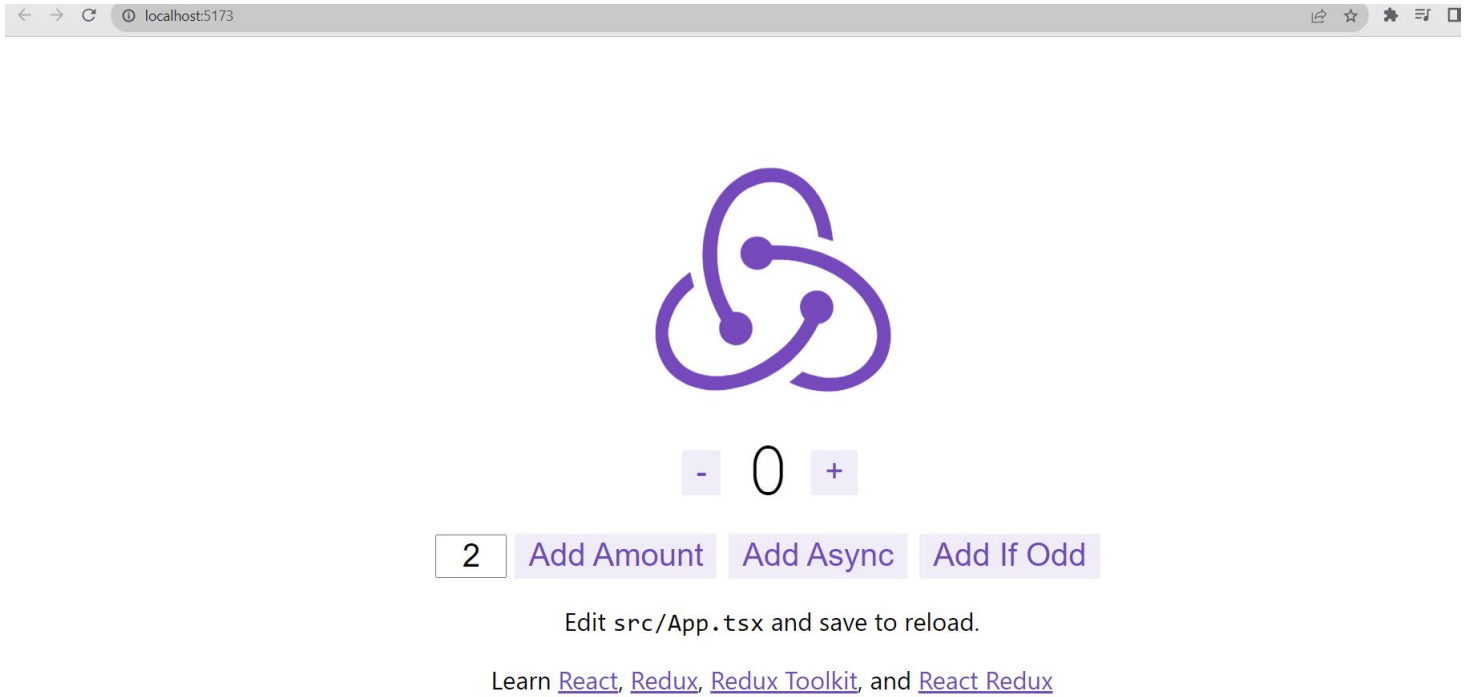
Устанавливаем все пакеты,
указанные в package.json с
помощью команды npm install.
Это может занять некоторое время



Установка

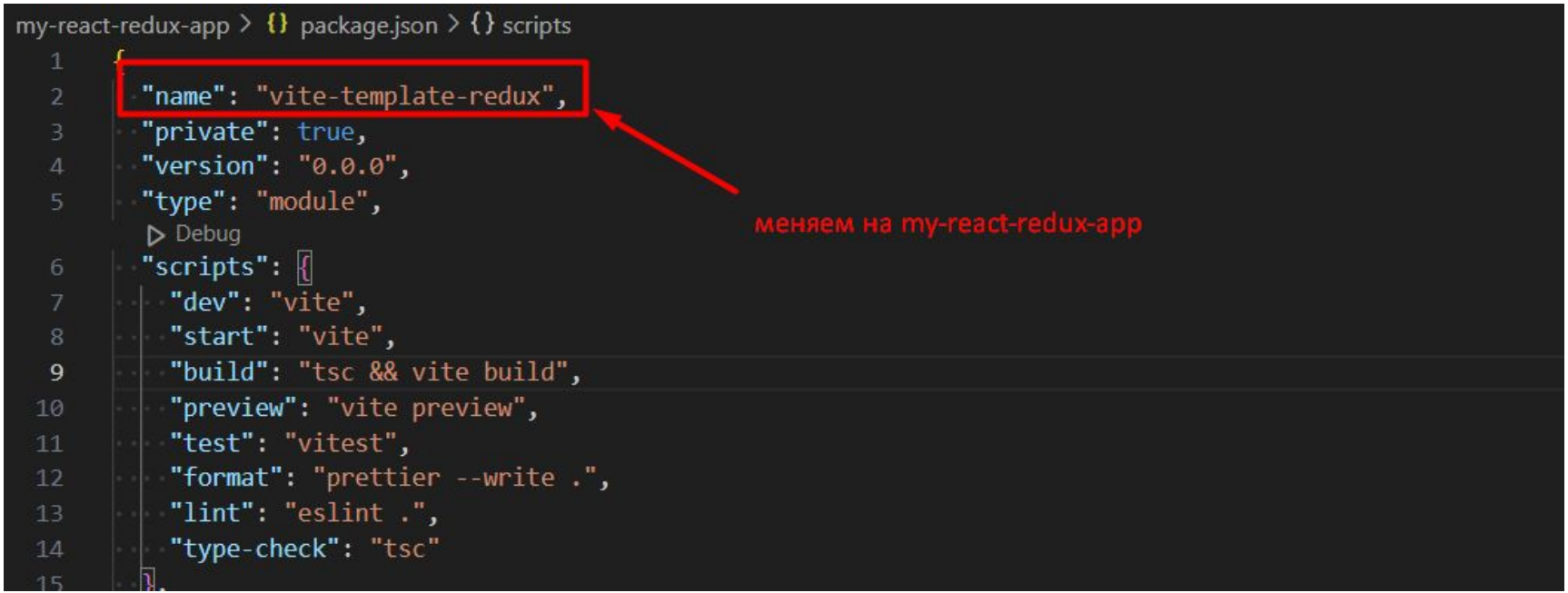
Шаг 5

Запускаем проект с помощью команды npm start. Как результат мы увидим уже созданный с помощью Redux Toolkit счётчик



Настройка

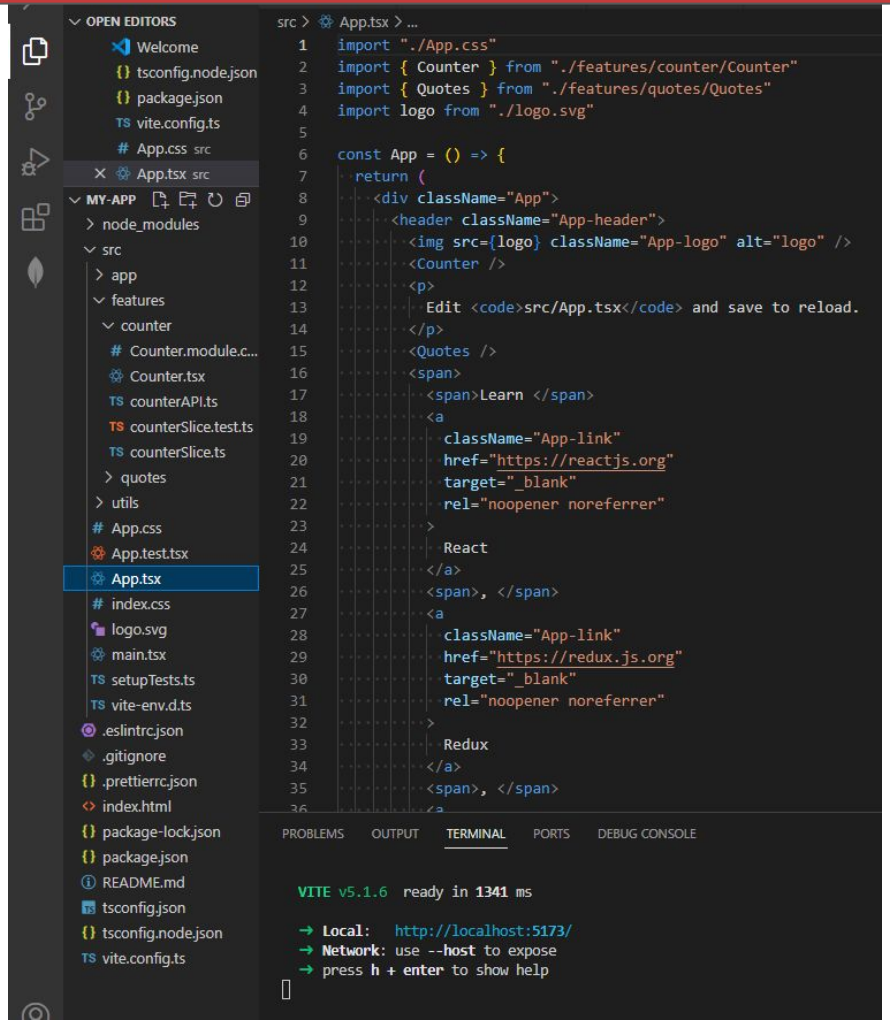
Открываем файл package.json и меняем название нашего проекта




```
{
  "name": "vite-template-redux",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  ▷ Debug
  "scripts": {
    "dev": "vite",
    "start": "vite",
    "build": "tsc && vite build",
    "preview": "vite preview",
    "test": "vitest",
    "format": "prettier --write .",
    "lint": "eslint .",
    "type-check": "tsc"
  },
```

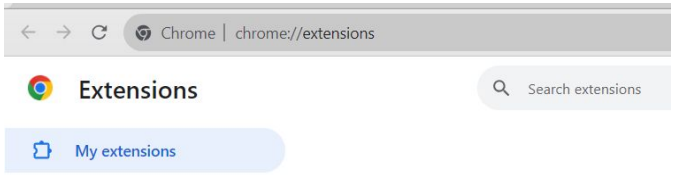
Структура проекта

- У нас появился новый файл с настройкой vite - vite.config.ts
- Созданный counter находится в папке features
- В папке app лежит файл с настройкой store и файл с вспомогательными хуками hooks.ts

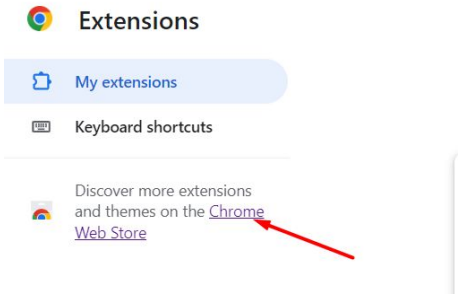


Установка расширения для браузера Redux dev tools

1. Перейдите на страницу расширений в Chrome.



2. Перейдите в раздел поиска расширений



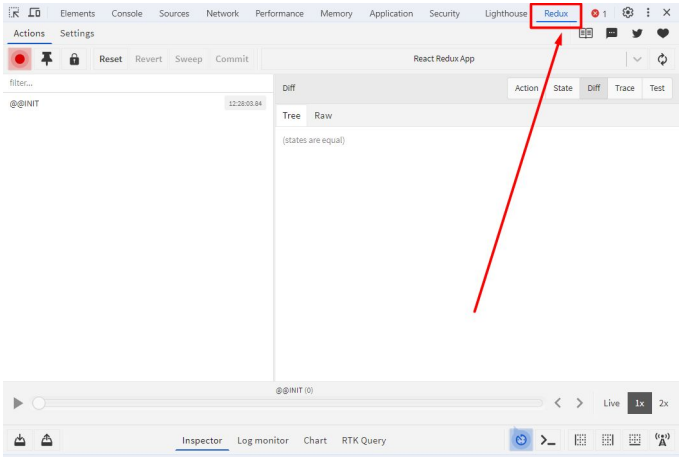
Установка расширения для браузера Redux dev tools

3. Найдите и установите расширение Redux DevTools

1) Redux DevTools Extension

2) Add to Chrome

4. Установленное расширение будет отображаться в инспекторе в браузере (F12)



Redux introduction

Рассмотрим основные понятия и схему работы Redux

Инициализация:

- Хранилище (Store): Здесь хранится все состояние приложения, например, счет в игре.

Действие (Action):

- Пользователь зарабатывает очко, и это событие называется "Увеличить счет".

Диспатч (Dispatch):

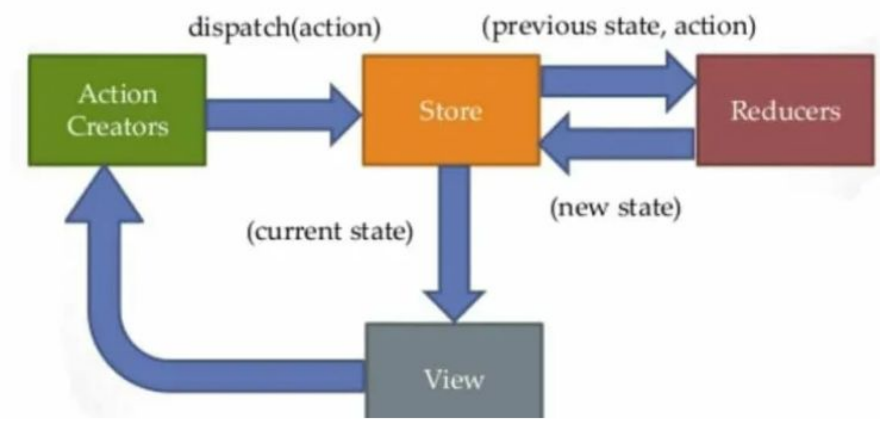
- Диспатч отправляет действие "Увеличить счет" в хранилище.

Редьюсер (Reducer):

- Редьюсер обрабатывает действие и обновляет состояние хранилища. Например, увеличит текущий счет.

Подписчик (Subscriber):

- Подписчик отслеживает изменения в хранилище. Если счет увеличился, подписчик может обновить интерфейс, чтобы отразить новый счет.



Предметный пример работы Redux

Давайте представим, что ваше веб-приложение — это большая коробка с игрушками. Каждая игрушка — это какая-то информация или часть вашего приложения.

- **Хранилище** (Redux Store): Это место, где вы кладете все свои игрушки, чтобы они были в одном месте и легко доступны.
- **Действия** (Actions): Это какие-то инструкции, чтобы что-то сделать с игрушками. Например, "Добавь новую игрушку" или "Измени цвет этой игрушки".
- **Редьюсер** (Reducer): Это специальные инструкции, как изменить игрушки в ответ на действия. Если вы получаете инструкцию "Добавь новую игрушку", редьюсер знает, как это сделать.
- **Диспатч** (Dispatch): Это как почтовая служба, которая разносит ваши инструкции (действия) редукторам (инструкциям по изменению игрушек).
- **Подписчики** (Subscribers): Это люди, которые следят за изменениями в коробке с игрушками. Если что-то меняется (например, добавляется новая игрушка), они могут быстро узнать и обновить информацию.

Зачем нужен Redux?

- **Упрощение управления состоянием:** Когда ваши приложения становятся сложными и содержат большое количество состояния (например, данные пользователя, настройки, текущее состояние интерфейса и др.), управление этим состоянием может стать сложной задачей. Redux предоставляет паттерн и инструменты для более предсказуемого и управляемого управления состоянием приложения.
- **Централизация состояния**: Redux использует одно центральное хранилище, где хранится весь стейт вашего приложения.
- **Предсказуемость изменений**: Состояние в Redux изменяется только через действия (actions), и эти изменения обрабатываются редьюсерами (reducers)
- **Легкость отладки и тестирования:** Redux обеспечивает простоту отслеживания изменений и, соответственно, тестирования

Настройка Redux Toolkit для React проекта

1. Создание slice (в файле counterSlice.ts) для управления состоянием счётчика. Slice - это часть хранилища, включающая в себя редьюсер, который управляет некоторой частью состояния, а также связанные с этим редьюсером действия.

```
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
import { createAppSlice } from "../../app/createAppSlice"

interface CounterState {
  value: number;
}

const initialState: CounterState = { value: 0 };

export const counterSlice = createAppSlice
({
  name: 'counter',
  initialState,
  reducers: create => ({
    increment: create.reducer(state => { state.value += 1; }),
    decrement: create.reducer(state => { state.value -= 1; }),
    incrementByAmount: create.reducer((state, action: PayloadAction<number>) => {
state.value += action.payload; }),
  }),
  selectors: {
    selectCount: counter => counter.value,
  },
});
export const { increment, decrement, incrementByAmount } = counterSlice.actions;
export const { selectCount} = counterSlice.selectors
```

Настройка Redux Toolkit для React проекта

2. Создание файла **store.ts** для настройки хранилища

```
import type { Action, ThunkAction } from "@reduxjs/toolkit"
import { combineSlices, configureStore } from "@reduxjs/toolkit"
import { counterSlice } from "../features/counter/counterSlice"

const rootReducer = combineSlices(counterSlice)
export type RootState = ReturnType<typeof rootReducer>

// The store setup is wrapped in `makeStore` to allow reuse
// when setting up tests that need the same store config
export const makeStore = (preloadedState?: Partial<RootState>) => {
  const store = configureStore({
    reducer: rootReducer,
    preloadedState,
  })
  return store
}

export const store = makeStore()
```

продолжение

Настройка Redux Toolkit для React проекта

3. Теперь мы можем использовать счётчик в нашем компоненте. Например, в файле Counter.tsx:

```
import { useState } from "react"
import { useDispatch, useSelector } from "react-redux"
import styles from "./Counter.module.css"
import {
  decrement,
  increment,
  incrementByAmount,
  selectCount
} from "./counterSlice"

function Counter (){
  const dispatch = useDispatch()
  const count = useSelector(selectCount)
  const [incrementAmount, setIncrementAmount] = useState<number>(0)

  const incrementValue = Number(incrementAmount) || 0
```


продолжение

Настройка Redux Toolkit для React проекта

4. Теперь мы можем использовать счётчик в нашем компоненте. Например, в файле Counter.tsx:

```
  return (
    <div>
      <div>
        <button onClick={() => dispatch(decrement())}>-</button>
        <span>{count}</span>
        <button onClick={() => dispatch(increment())}>+</button>
      </div>
      <div>
        <input
          value={incrementAmount}
          type="number"
          onChange={e => {
            setIncrementAmount(e.target.value)
          }}
        />
        <button onClick={() => dispatch(incrementByAmount(incrementValue))}>
          Add Amount
        </button>
      </div>
    </div>
  )
}
```

Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH

AIT
Hi-tech team