React: useContext



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;)

В чем разница SPA и MPA?

Какие типы ссылок есть в react router?

Какие вспомогательные хуки в react router вы знаете?

Для чего используется useFormik?

Как выполнить действие при размонтировании компонента?

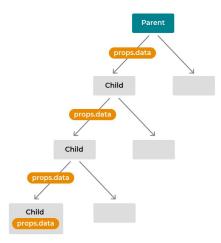
ЦЕЛЬ

Изучить способ передачи данных через большую вложенность компонентов

ПЛАН ЗАНЯТИЯ

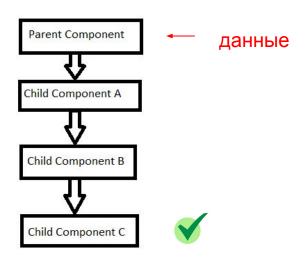
- 1. Props drilling
- 2. useContext
- 3. Установка Vite проекта

useContext



useContext - это React Hook, предоставляющий способ получения значения из контекста в функциональных компонентах React.

Контекст в React используется для передачи данных глубоко вложенным компонентам без явной передачи пропс через каждый уровень.



Шаг 1. Создаём контекст в компоненте, который будет передавать значение

```
import { createContext, useState } from 'react';
// Создаем контекст с типом React.Context и значением по умолчанию
const MyContext = createContext<string>("");
                                                                            создаём контекст
// Компонент-поставщик контекста
const MyContextProvider = ({ children }) => {
  const [sharedValue, setSharedValue] = useState('Значение из контекста'
  const updateValue = (newValue) => {
                                                                            делаем компонент
    setSharedValue(newValue);
                                                                            поставщиком
  };
                                                                            контекста
  return (
    <MyContext.Provider value={{ sharedValue, updateValue }}>
      {children}
    </MyContext.Provider>
  );
Э;
export { MyContext, MyContextProvider };
```

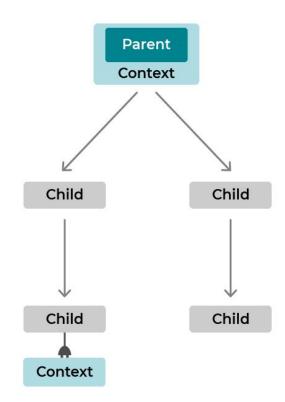
Шаг 2. Соответствующий <Context.Provider> должен находиться выше компонента, которому нужны данные из контекста. В данном случае **MyContextProvider** находится на самом верхнем уровне, но это необязательно приложения

```
import React from 'react';
                                                                        импортируем
import { MyContextProvider } from './MyContextProvider'
                                                                        компонент с
import MyComponent from './MyComponent';
                                                                        контекстом
const App = () => {
 return
                                                                  Оборачиваем другие
    <MyContextProvider>
                                                                  компоненты
      <div>
        <h1>Moe React Приложение</h1>
        <MyComponent />
        {/* Другие компоненты в приложении */}
      </div>
    </MyContextProvider>
  );
};
export default App:
```

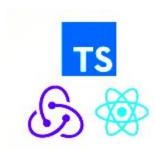
Шаг 3. Передаём значение из компонента поставщика (провайдера) в другой компонент

```
import React, { useContext } from 'react';
import { MyContext } from './MyContextProvider'; <--</pre>
                                                                                           импортируем
                                                                                           контекст
// Компонент, использующий значение из контекста с помощью useContext
const MyComponent = () => {
  const { sharedValue, updateValue } = useContext(MyContext);
                                                                                    С помощью useContext
                                                                                    получаем доступ к
  const handleButtonClick = () => {
                                                                                    значению и функции
    // Обновляем значение в контексте
                                                                                    для его изменения из
   updateValue('Новое значение из компонента');
                                                                                    компонента
  3;
                                                                                    MyContextProvider
  return (
    <div>
      >Значение из контекста: {sharedValue}
      <button onClick={handleButtonClick}>Обновить значение/button>
    </div>
  );
3;
export default MyComponent;
```

Когда значение в контексте обновляется в MyContextProvider, все компоненты, использующие useContext(MyContext), автоматически получают новое значение, и их перерисовывается.



Создание проекта React Redux



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

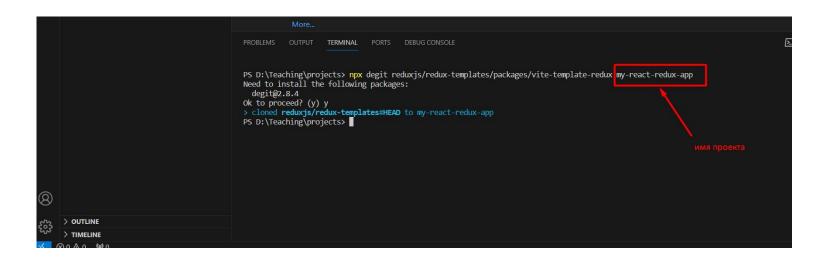
Примечание: Vite — это инструмент сборки, цель которого — обеспечить более быструю и экономичную разработку современных веб-проектов.



Шаг 1

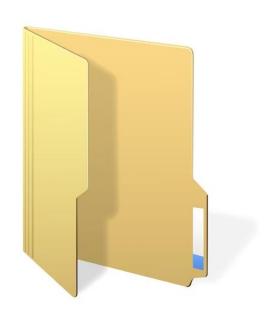
Откройте VSCode, перейдите в папку, в которой будет лежать ваш проект. В терминале введите следующую команду

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



Шаг 3

Открываем файл package.json и меняем название нашего проекта

```
my-react-redux-app > {} package.json > {} scripts
        "name": "vite-template-redux",
         "private": true,
         "version": "0.0.0",
         "type": "module",
         ▶ Debug
         "scripts": {
          "dev": "vite",
           "start": "vite",
           "build": "tsc && vite build",
  9
           "preview": "vite preview",
           "test": "vitest",
           "format": "prettier --write .",
           "lint": "eslint .",
           "type-check": "tsc"
```

Шаг 4

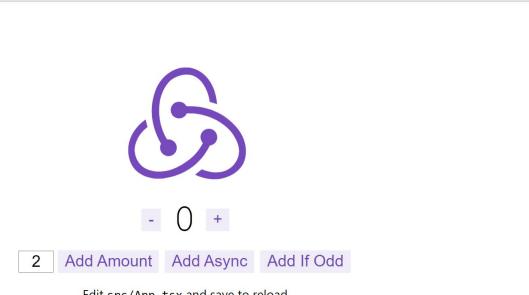
Устанавливаем все пакеты, указанные в package.json с помощью команды npm install. Это может занять некоторое время

```
y-react-redux-app > {} package.json > {} devDependencies
      "name": "my-react-redux-app",
       "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"
       "dependencies": {
        "@reduxjs/toolkit": "^1.8.1",
        "react": "^18.2.0",
        "react-dom": "^18.2.0",
        "react-redux": "^8.0.1"
       "devDependencies":
        "@testing-library/dom": "^9.2.0",
        "@testing-library/jest-dom": "^5.11.4",
        "@testing-library/react": "^14.0.0",
        "@testing-library/user-event": "^14.2.5",
        "@types/react": "^18.0.15",
        "@types/react-dom": "^18.0.6",
        "@types/testing-library jest-dom": "^5.14.5",
        "@vitejs/plugin-react": "^4.0.0",
        "eslint": "^8.0.0",
        "eslint-config-react-app": "^7.0.1",
        "eslint-plugin-prettier": "^4.2.1",
        "jsdom": "^21.1.0",
```

Шаг 5

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

C O localhost:5173



应☆ ★ 司 □

Edit src/App.tsx and save to reload.

Learn React, Redux, Redux Toolkit, and React Redux

Настройка

Открываем файл package.json и меняем название нашего проекта

```
my-react-redux-app > {} package.json > {} scripts
        "name": "vite-template-redux",
         "private": true,
         "version": "0.0.0",
         "type": "module",
         ▶ Debug
         "scripts": {
          "dev": "vite",
          "start": "vite",
           "build": "tsc && vite build",
  9
           "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 арр лежит файл с настройкой store и файл с вспомогательными хуками hooks.ts

```
V OPEN EDITORS
                               src > 🏶 App.tsx > ...
                                       import "./App.css"
Ф
           ★ Welcome
                                       import { Counter } from "./features/counter/Counter"
           {} tsconfig.node.ison
                                       import { Quotes } from "./features/quotes/Quotes"
           {} package.json
                                       import logo from "./logo.svg"
           TS vite.config.ts
            # App.css src
                                       const App = () => {
         X 🎡 App.tsx src
      V MY-APP C C ひ 回
                                            <div className="App">
                                              <header className="App-header">
        > node modules
                                                <img src={logo} className="App-logo" alt="logo" />
       V STC
         > app
         features
                                                 Edit <code>src/App.tsx</code> and save to reload.
         v counter
          # Counter.module.c...
          Counter.tsx
                                                  <span>Learn </span>
          TS counterAPI.ts
          TS counterSlice.test.ts
                                                    className="App-link"
          TS counterSlice.ts
                                                    href="https://reactjs.org"
         > quotes
                                                    target=" blank"
         > utils
                                                    rel="noopener noreferrer"
         # App.css
                                                    React
        App.test.tsx
          App.tsx
         # index.css
        logo.svq
                                                    className="App-link"
        main.tsx
                                                    href="https://redux.js.org"
                                                    target=" blank"
        TS setupTests.ts
                                                    rel="noopener noreferrer"
        TS vite-env.d.ts
       eslintrc.json
                                                    Redux
       .gitignore
       {} .prettierrc.json
       index.html
       {} package-lock.json
                                PROBLEMS
                                                    TERMINAL
                                                                       DEBUG CONSOLE
       [] package.json
       (i) README.md
                                  VITE v5.1.6 ready in 1341 ms
       tsconfig.json
                                  → Local: http://localhost:5173/
       {} tsconfig.node.json
                                  → Network: use --host to expose
       TS vite.config.ts
                                  → press h + enter to show help
```



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в немецкой школе AIT TR GmbH

