React-Router-DOM (маршрутизация)

React Router с TypeScript

React Router DOM - это стандартная библиотека для маршрутизации в React приложениях. Она позволяет создавать одностраничные приложения (SPA) с навигацией между различными компонентами без перезагрузки страницы.

Основные концепции

1. Маршрутизация (Routing)

Маршрутизация - это процесс определения того, какой компонент должен быть отображен в зависимости от текущего URL.

2. Основные компоненты React Router DOM

- **Router** - обертка для всего приложения
- **Routes** - контейнер для маршрутов
- **Route** - определяет маршрут
- **Link** - для навигации между страницами
- **Navigate** - для программного перенаправления

Установка

```
npm install react-router-dom
npm install --save-dev @types/react @types/react-dom
```

Базовая настройка

App.tsx - Основной файл приложения

tsx

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-route
r-dom';

import HomePage from './pages/HomePage';
import AboutPage from './pages/AboutPage';
import UsersPage from './pages/UsersPage';
import UserDetailPage from './pages/UserDetailPage';
import NotFoundPage from './pages/NotFoundPage';

const App: React.FC = () => {
  return (
    <Router>
      <div>
        <Routes>
          <Route path="/" element={<HomePage />} />
          <Route path="/about" element={<AboutPage />} />
          <Route path="/users" element={<UsersPage />} />
          <Route path="/users/:id" element={<UserDetailPage />} />
          <Route path="/old-about" element={<Navigate to="/about" replace />}
/>
          <Route path="*" element={<NotFoundPage />} />
        </Routes>
      </div>
    </Router>
  );
};

export default App;
```

Объяснение структуры:

1. **Router** - оборачивает все приложение и обеспечивает контекст маршрутизации
2. **Routes** - содержит все маршруты приложения
3. **Route** - определяет отдельный маршрут с путем и компонентом

Типы маршрутов

1. Статические маршруты

```
<Route path="/" element={<HomePage />} />
<Route path="/about" element={<AboutPage />} />
```

2. Динамические маршруты с параметрами

```
<Route path="/users/:id" element={<UserDetailPage />} />
```

`:id` - параметр маршрута, который можно получить в компоненте

3. Перенаправления

```
<Route path="/old-about" element={<Navigate to="/about" replace />} />
```

4. Catch-all маршрут (404)

```
<Route path="*" element={<NotFoundPage />} />
```

Навигация

1. Декларативная навигация с Link

```
// Navigation.tsx
import React from 'react';
```

```
import { Link } from 'react-router-dom';

const Navigation: React.FC = () => {
  return (
    <nav>
      <Link to="/" style={{ marginRight: '20px' }}>Главная</Link>
      <Link to="/about" style={{ marginRight: '20px' }}>О нас</Link>
      <Link to="/users" style={{ marginRight: '20px' }}>Пользователи</Link>
    </nav>
  );
};

export default Navigation;
```

2. Программная навигация с useNavigate

```
// AboutPage.tsx
import React from 'react';
import { useNavigate } from 'react-router-dom';

const AboutPage: React.FC = () => {
    const navigate = useNavigate();

    return (
        <div>
            <h1>О нас</h1>
            <button onClick={() => navigate('/')}>
                Вернуться на главную (программно)
            </button>
            <button onClick={() => navigate(-1)}>
                Назад
            </button>
        </div>
```

```
    );
};
```

Работа с параметрами маршрутов

useParams - получение параметров из URL

```
// UserDetailPage.tsx
import React from 'react';
import { useParams, useNavigate, Link } from 'react-router-dom';
import { users } from '../shared/mockData/users';

const UserDetailPage: React.FC = () => {
    const { id } = useParams<{ id: string }>();
    const navigate = useNavigate();

    const user = users.find(u => u.id === Number(id));

    if (!user) {
        return (
            <div>
                <h1>Пользователь не найден</h1>
                <Link to="/users">К списку пользователей</Link>
            </div>
        );
    }

    return (
        <div>
            <h1>Профиль: {user.name}</h1>
            <p>Email: {user.email}</p>
            <p>ID: {user.id}</p>
            <button onClick={() => navigate('/users')}>К списку</button>
            <button onClick={() => navigate(-1)}>Назад</button>
        </div>
```

```
    );
};
```

Практические примеры

1. Список пользователей с навигацией

```
// UsersPage.tsx
import React from 'react';
import { Link } from 'react-router-dom';
import { users } from '../shared/mockData/users';

const UsersPage: React.FC = () => {
    return (
        <div>
            <h1>Пользователи</h1>
            {users.map(user => (
                <div key={user.id} className="user-card">
                    <h3>{user.name}</h3>
                    <p>{user.email}</p>
                    <Link to={`/users/${user.id}`}>
                        Подробнее
                    </Link>
                </div>
            ))}
        </div>
    );
};
```

2. Обработка 404 ошибок

```
// NotFoundPage.tsx
import React from 'react';
import { Link } from 'react-router-dom';
```

```
const NotFoundPage: React.FC = () => {
   return (
      <div>
         <h1>404 - Страница не найдена</h1>
         <Link to="/">Вернуться на главную</Link>
      </div>
   );
};
```

Типы данных (TypeScript)

User.ts

```
export interface User {
  id: number;
  name: string;
  email: string;
}
```

users.ts - Моковые данные

```
import type { User } from '../types/User';

export const users: User[] = [
  { id: 1, name: 'Алексей Иванов', email: 'alexey@example.com' },
  { id: 2, name: 'Мария Петрова', email: 'maria@example.com' },
  { id: 3, name: 'Дмитрий Сидоров', email: 'dmitry@example.com' },
];
```

Основные хуки React Router DOM

1. useNavigate()

- Программная навигация
- `navigate('/path')` - переход на страницу
- `navigate(-1)` - возврат назад
- `navigate(1)` - переход вперед

2. useParams()

- Получение параметров из URL
- Возвращает объект с параметрами

3. useLocation()

- Информация о текущем местоположении
- pathname, search, hash, state