Lesson 12 - Маршрутизация. React Router

Routing (маршрутизация, роутинг, раутинг)

Отличающее преимущество веб-приложения от десктоп-приложения это наличие URL, при переходе по которому, пользователь оказывается в определенной части приложения. Таким образом можно сохранить закладку или передать ссылку другому пользователю, при этом ему будет отображен один и тот же интерфейс (за исключением авторизации).

Роутинг - это не побочный эффект при написании приложения, наоборот, его структуру необходимо продумывать в первую очередь. Грамотная архитектура навигации определяет насколько быстро и удобно можно внедрять новый функционал.

Структруа URL-строки

Аналогией URL-строки может быть адрес по которому вы проживаете: улица, дом, квартира. Таким образом у каждого состояния нашего интерфейса должен быть свой адрес, свой URL.

То что видит пользователь, состояние интерфейса, должно быть описано в URL.

- **https://** - протокол
- **mysite.com/** - хост
- **books/e3q76gm9lzk** - путь, то где мы находимся в приложении
- **e3q76gm9lzk** - url-параметр. Параметры бывают динамическими или статическими
- **?category=adventure&status=unread** - строка запроса
- **?** - символ начала строки запроса
- **category=adventure** - пара параметр=значение
- **&** - символ "И", разделяет параметры строки запроса
- **#comments** - якорь, определяет положение на странице

history

История - то как мы переходим ссылкам, то как эти переходы хранятся и парсятся. В зависимости от типа истории зависит метод хранения истории и ее изменение.

- **Hash history** - в старых браузерах не поддерживается HTML5 history API, поэтому для них существует эта реализация истории.
- **Browser history** - использует HTML5 history API, стандарт управления историей браузера из JavaScript.
- **Memory history** - позволяет использовать историю сессии вне окна браузера. К примеру для

тестирования логики без интерфейса.

Особенность browserHistory

- При использовании такой истории, контент-сервер на любой запрос должен отдавать `index.html`, а управление маршрутизацией берет на себя React Router.
- Если используется кастомная сборка `webpack`, то для `devServer` необходимо указать свойство `historyApiFallback: true`

React Router

Предоставляет декларативный API (набор компонентов) для управления частями URL-строки и отображения различных компонетов в зависимости от текущего ее состояния.

Документация React Router

Разбит на пакеты для различных платформ, нас интересует `react-router-dom`.

```
npm install react-router-dom
```

или

```
yarn add react-router-dom
```

В React Router есть три типа компонентов: компонент маршрутизатора, компоненты согласования маршрутов и компоненты навигации.

BrowserRouter

В основе каждого одностраничного приложения стоит маршрутизатор.

BrowserRouter - компонент для создания роутера, использует HTML5-историю (pushState, replaceState и popstate), чтобы синхронизировать интерфейс с URL-адресом. Используя контекст, передает данные о текущем URL всему поддереву компонентов.

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

ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.querySelector('#root'),
);
```

Документация BrowserRouter

Route

Route - самый важный компонент. Его задача заключается в том, чтобы отобразить некоторый UI, когда `location.pathname` соответствует значению пропа `path`.

```
import React from 'react';
import { Route } from 'react-router-dom';

const App = () => (
  <div>
    <Route exact path="/" component={Home} />
    <Route path="/about" component={About} />
    <Route path="/contact" component={Contact} />
  </div>
);
```

Когда `location.pathname === '/about'`, первый и третий рауты отрендерят `null`, а второй отрендерит компонент `About`.

- Согласование маршрута выполняется путем сопоставления пропа `path` и текущего значения `location.pathname`.
- Если значение `location.pathname` начинается на указанный пусть в `path`, `<Route>` отрендерит указанный компонент, в противном случае отрендерит `null`.
- `<Route>` без указанного пропа `path` всегда рендерит компонент.
- `<Route>` можно использовать в любом месте где необходимо отображать контент на основе текущего местоположения.
- Проп `exact` указывает на необходимость точного совпадения пропа `path` и `location.pathname`.

Документация Route

Switch

Switch - используется для группировки `<Route>`. Отображает первый дочерний `<Route>`, `path` которого соответствует текущему `location.pathname`.

```
import React from 'react';
import { Switch, Route } from 'react-router-dom';

const App = () => (
  <div>
    <Switch>
      <Route exact path="/" component={Home} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Route component={PageNotFound} />
    </Switch>
  </div>
```

```
);
```

Redirect

Позволяет принудительно перенаправить пользователя на другой маршрут.

```
import React from 'react';
import { Switch, Route, Redirect } from 'react-router-dom';

const App = () => (
  <div>
    <Switch>
      <Route exact path="/" component={Home} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
      <Redirect to="/" />
    </Switch>
  </div>
);
```

Документация Redirect

Link и NavLink

React Router предоставляет декларативную навигацию в приложении используя компоненты `<Link>` и `<NavLink>` для создания ссылок.

```
<Link to="/books">Books</Link>

<Link to="/books?category=adventure#treasure-island">Adventure Books</Link>
```

Проп `to` можно передавать в виде строки описывающей `href` будущей ссылки.

Или как объект со следующими (необязательными) свойствами:

- **pathname** - строка, путь для ссылки
- **search** - строковое представление параметров запроса
- **hash** - хэш для добавления в конец URL
- **state** - объект, который будет записан в `location.state` после перехода по ссылке

```
<Link
  to={{
    pathname: '/books',
    search: '?category=adventure',
    hash: '#treasure-island',
    state: { from: '/dashboard' },
  }}
/>
```

`<NavLink>` - особый тип `<Link>`, который может иметь дополнительные стили, если текущий URL совпадает со значением пропа `to`.

- **activeClassName** - строка классов для объеденения с `className` когда элемент активен.
- **activeStyle** - объект инлайн стилей для добавления к элементу когда он активен.
- **exact** - когда `true`, активные классы/стили будут применяться только в том случае, если местоположение точно совпадает со значением пропа `to`.

```
<NavLink to="/books" activeClassName="active">
  Books
</NavLink>
```

- Документация Link
- Документация NavLink

Route в деталях

Есть 3 способа (пропа) отрендерить UI используя `<Route>`: `component`, `render` и `children`

- Каждый из них имеет свое применение.
- В одном `<Router>` можно использовать только один из них.
- Большую часть времени используется `component`.

component

- Используется когда есть готовый компонент и его необходимо отобразить без передачи дополнительных `props`.
- Для создания компонента используется `React.createElement`.

```
// Хорошо
<Route path="/about" component={About} />

// Так НИКОГДА не делать!!!
<Route path="/about" component={props => <About {...props} extraProp="amazing prop"
/>} />
```

Использование `React.createElement` означает, что если вместо ссылки на компонент будет передана анонимная функция, для каждого ре-рендера будет создан новый компонент. Это приведет к размонтированию существующего компонента и добавлению нового, вместо обновления существующего компонента.

render

```
// Хорошо
<Route path="/about" render={() => <div>About Page</div>} />
```

```
// Хорошо
<Route path="/about" render={props => <About {...props} extraProp="amazing prop"
/>} />
```

- Позволяет использовать инлайн-функцию вместо компонента без нежелательного эффекта ре-рендера как в случае с `component`.
- Используется тогда, когда компоненту необходимо передать дополнительные пропы.

Вместо того, чтобы создать новый React-элемент, переданнвя функция будет вызываться когда `location.pathname` и `path` совпадают.

Route props

Как бы не маршрут не ренделил компонент, ему передаются специальные пропы `match`, `location` и `history`.

match

Объект, содержит информацию о том, как `path` совпал с URL. Содержат следующие свойства:

- **params** - объект пар `ключ:значение`, соответствующих динамическим параметрам URL
- **isExact** - буль, указыват на точное соотвествие `path` и URL
- **path** - строка, паттерн пути на который замачился `<Route>`. Используется для создания вложенных маршрутов.
- **url** - строка, совпавшая часть URL-адреса. Используется для создания вложенной навигации.

location

Объект, поля которого описывают текущее местоположение, пусть куда будет произведен переход и откуда пришли на текущий маршрут.

```
{
  key: 'ac3df4',
  pathname: '/books'
  search: '?sortby=latest',
  hash: '#comments',
  state: {
    from: '/login'
  }
}
```

Документация Route props