1. Если в этом примере, изменить порядок Route таким образом:

```
<Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About/>} />
        <Route path="/users" element={<Users />} />
</Routes>
```

То какой компонент будет отрисован по адресу /users? Объясните почему.

Ответ:

По адресу /users будет отрисован компонент <Users />.

Это происходит потому, что внутри компонента <Routes> мы определяем маршруты с помощью компонента <Route>. У каждого маршрута есть свойство path, которое указывает URL-адрес, на который он должен соответствовать, и свойство element, которое определяет компонент, который должен быть отрисован, когда маршрут соответствует данному URL-адресу.

В данном случае мы определяем три маршрута: первый соответствует URL-адресу /, второй - /about, а третий - /users. Когда пользователь переходит по URL-адресу /users, React Router сопоставляет этот адрес со всеми маршрутами в <Routes>. Он находит маршрут, который соответствует URL-адресу /users, и отрисовывает компонент, который был указан в свойстве element этого маршрута, в данном случае - <Users />.

2. Какое сообщение появится на экране по адресу http://localhost:3000/users/12345

```
...
<Route path='/users/:number' element={<User />} />
...

function User(props) {
   const number = props.match.params.number;
   return(
      number===12345
         ? <h1>Моя личная страница</h1>
         : <h1>Страница пользователя {number}</h1>
   );
}
```

Ответ:

“**Моя личная страница**”

3. Вспомните, какой второй параметр принимает метод `map`

Ответ:

Метод map() принимает второй необязательный параметр, который называется thisArg. Этот параметр определяет значение this, которое будет использоваться при вызове функции обратного вызова внутри метода map(). Если он не указан, то this будет иметь значение undefined. Если параметр thisArg указан, то функция обратного вызова будет вызвана с указанным значением this. Это может быть полезно, если вы хотите использовать метод map() с методом объекта, который использует this.

4. Как бы вы подошли к решению задачи по выводу компонентом `<CardList>` только тех экземпляров компонента `<Card>`, цена которых не превышает заданную?

Ответ:

Можно решить эту задачу используя метод filter для отбора тех товаров, цена которых не превышает заданную, и затем передавать только эти товары в компонент <CardList>

5. Как задать параметр в пути? Например, *filter (*`useLocation, useSearchParams`*)*

Ответ:

Для задания параметра в пути в React Router вы можете использовать объект history и его метод push. мы создаем объект history с помощью хука useHistory(). Затем мы создаем функцию handleClick, которая использует метод push объекта history, чтобы перейти на URL-адрес /my-path с параметром filter и его значением myFilterValue. Мы связываем эту функцию с обработчиком события onClick кнопки.

Когда пользователь кликает на эту кнопку, React Router перенаправляет его на указанный URL-адрес с заданным параметром в пути. Вы можете получить значение параметра в компоненте, связанном с этим URL-адресом, с помощью хука useLocation() и объекта location.search.

6. Какая разница между `element` и `children` в указании роутера?

Ответ:

element используется для передачи React-компонента в качестве дочернего компонента, который будет отображаться при сопоставлении маршрута, а children используется для передачи дочерних компонентов напрямую внутри компонента Route. Также, при использовании `element`, компонент, указанный в качестве значения `element`, получит дополнительные пропсы, такие как `location`, `match` и `history`. При использовании `children`, компонент, возвращаемый функцией, не получит эти пропсы автоматически.

7. Зачем нужен `exact`?

Ответ:

Атрибут exact используется в React Router для сопоставления пути точно с тем, который указан в атрибуте path.

8. Самостоятельно разберитесь, зачем нужны `useMatch`, `useLocation`, `useNavigate`?

Ответ:

useMatch, useLocation, и useNavigate - это React Hook'и, предоставляемые React Router.

- useMatch используется для получения информации о сопоставлении текущего маршрута и предоставленного шаблона пути. С помощью этого хука можно получить параметры, переданные в адресной строке.
- useLocation используется для получения информации о текущем URL и его параметрах. Это позволяет компонентам получать и использовать информацию о текущем пути и переданных параметрах.
- useNavigate используется для перехода на другую страницу в приложении. Он позволяет программно управлять маршрутизацией в React Router.

Все эти хуки позволяют разработчикам более гибко управлять маршрутизацией в React приложениях и получать доступ к информации о текущем пути и его параметрах.

9. Как можно сделать перенаправление на другую страницу по клику на кнопку с помощью `useNavigate`? *Ищите ответ в документации к react-router-dom*

Для перенаправления на другую страницу по клику на кнопку с помощью useNavigate нужно сделать следующее:

- Импортировать useNavigate из библиотеки react-router-dom:

import { useNavigate } from ‘react-router-dom';

- Использовать useNavigate внутри компонента, например, в обработчике события клика на кнопке:

```
function MyComponent() {
  const navigate = useNavigate();

  const handleClick = () => {
    navigate('/new-page');
  };

  return (
    <button onClick={handleClick}>Перейти на новую страницу</button>
  );
}
```

В этом примере, при клике на кнопку будет вызван обработчик handleClick, который вызовет функцию navigate с параметром в виде пути к новой странице - /new-page. После этого произойдет перенаправление на указанную страницу.