# Лекция 11
# Расширенные возможности React Router

Изучите возможности использования строк запроса для обмена и управления состоянием приложения через URL-адреса, работайте с различными типами данных в параметрах запроса и используйте React Router для навигации и эффектов в зависимости от изменений URL-адреса.

В этой лекции мы углубимся в обработку параметров запроса в React Router. Ранее мы рассмотрели теорию работы с URL-адресами и способы обработки путей с помощью React Router. Теперь мы готовы изучить мир параметров запроса.

React Router предоставляет нам инструменты для эффективной работы с параметрами запросов. Мы предоставим практический опыт, который позволит вам получить полезные знания. Мы добавим еще одну страницу в существующее приложение, что позволит нам работать со строками запросов. К концу этого раздела вы сможете беспрепятственно использовать параметры запросов в своих приложениях, работающих на основе React Router.

## 1. Понимание строк запроса в URL-адресах

Строки запроса — это способ передачи информации между веб-страницами и приложениями, часто используемый для фильтрации, поиска или персонализации пользовательского интерфейса. Извлекая данные из строк запроса, веб-приложения могут корректировать свое поведение в зависимости от ввода пользователя или другой контекстной информации, предоставляемой через URL-адрес.

**Пример**. Рассмотрим простой пример URL-адреса с поисковой строкой:

```
https://example.com/products?category=electronics&price=100
```

В этом URL:

– Базовый URL-адрес: https://example.com/products;
– Строка запроса начинается с вопросительного знака ?;
– Имеется два параметра: category и price;
– Соответствующие значения этих параметров — electronics и 100.

Когда пользователь переходит по этому URL-адресу, веб-приложение может прочитать и интерпретировать строку запроса. Например, оно может отфильтровать товары по категории «электроника» и отобразить товары по цене 100 долларов или ниже. Это позволяет пользователям напрямую получать доступ к определенному контенту или пользовательским представлениям веб-сайта.

## 2. Настройка компонента React

Наша следующая задача в рамках работы над предыдущим приложением — создание страницы «Продукты». Эта страница послужит платформой для отображения данных, предоставленных из файла **data.js**, что позволит нам взаимодействовать с ними.

Начнём с создания ProductsPage компонента. Структура компонента выглядит следующим образом:

```
import React, { useState } from "react";

const ProductsPage = ({ data }) => {
  const [companies, setCompanies] = useState(data);

  const handleChange = (e) => {
    // Handle user input and update the state (companies in this case)
  };

  return (
    <>
      <input type="text" onChange={handleChange} />
      <ul className="companyList">
        {companies.map(({ id, companyName, companyDescription }) => (
          <li key={id}>
            <h2>{companyName}</h2>
            <p>{companyDescription}</p>
          </li>
        ))}
      </ul>
    </>
  );
};

export default ProductsPage;
```

– Мы создали компонент React под названием ProductsPage. Этот компонент предназначен для отображения списка компаний.

– Этот useState хук инициализирует companies состояние. Это состояние позволит компоненту управлять изменениями в списке компаний и реагировать на них;

– Данная handleChange функция предназначена для обработки пользовательского ввода и запуска обновлений на основе этих данных.

Благодаря этому ProductsPage компоненту мы можем отображать динамический список компаний и предоставлять пользователям возможность взаимодействовать с данными.

## 3. Анализ последних изменений

Давайте на мгновение остановимся и рассмотрим недавние изменения в нашем приложении:

– **Новый маршрут**: Мы добавили /products маршрут для App.jsx создания страницы «Продукты», расширив тем самым функциональность приложения;

– **Улучшенная навигация**: теперь навигация **Bar.jsx** включает ссылку на страницу «Продукты», что упрощает пользователям доступ к этому новому разделу;

– ProductsPage **Компонент**: Сердцем нашей страницы «Товары» является ProductsPage компонент. Он отображает и управляет данными о товарах, обеспечивая интерактивный интерфейс.

Изучите все возможности приложения в предоставленной среде разработки Code Sandbox, чтобы увидеть эти обновления в действии.

URL:https://codesandbox.io/p/sandbox/09f4a42b-8966-4e17-bc0b-f65c1cc14046-g2lypv?from-embed

## 4. Работа с параметрами запроса

Это useSearchParams хук, предоставляющий доступ к параметрам запроса (параметрам поиска) текущего URL.

Параметры запроса представляют собой пары ключ-значение, следующие за вопросительным знаком (?) в URL. Например, в URL http://example.com/products?category=electronics&page=2 параметры запроса — это category=electronics и page=2.

**Что возвращает функция useSearchParams**?

Хук useSearchParams возвращает массив с двумя элементами: объектом, представляющим текущие параметры запроса, и функцией для обновления параметров запроса. Вот как можно использовать хук:

```
import { useSearchParams } from "react-router-dom";

function MyComponent() {
  const [searchParams, setSearchParams] = useSearchParams();

  // `searchParams` is an object representing the current query parameters
  // `setSearchParams` is a function to update the query parameters
}
```

**Объект searchParams**

Объект searchParams является экземпляром класса URLSearchParams. Он предоставляет удобные методы для взаимодействия с параметрами запроса, такие как:

– get(key) Извлекает значение, связанное с определенным ключом;

– getAll(key) Извлекает все значения, связанные с определенным ключом;

– set(key, value): Устанавливает или обновляет значение определенного ключа;

– append(key, value ) Добавляет новое значение к существующему ключу;

– delete(key) Удаляет определенный ключ и связанное с ним значение;

– toString(): Возвращает параметры запроса в виде строки (например, "?category=electronics&page=2").

**Функция setSearchParams**

Эта setSearchParams функция позволяет обновлять параметры запроса в URL. Вы можете передать ей объект с парами ключ-значение для установки новых параметров или изменения существующих. Вот пример использования:

```
// Suppose the current URL is
`http://example.com/products?category=electronics&page=2`

setSearchParams({ category: "clothing", color: "blue" });
// The URL is now `http://example.com/products?category=clothing&color=blue`
```

## 5. Интеграция хука useSearchParams

В примере компонента ProductsPage мы интегрируем useSearchParams функцию управления параметрами запроса.

```
import React, { useState, useEffect } from "react";
import { useSearchParams } from "react-router-dom";

const ProductsPage = ({ data }) => {
  const [companies, setCompanies] = useState(data);
  const [searchParams, setSearchParams] = useSearchParams();

  // Function to update the displayed `companies` based on query parameters
  const updateCompanies = (params) => {
    const filteredCompanies = data.filter((item) => {
      const { companyName } = item;
      if (params.get("name")) {
        return companyName.toLowerCase().includes(params.get("name"));
      }
      return true;
    });
    setCompanies(filteredCompanies);
  };

  useEffect(() => {
    // When the component mounts or query parameters change, update the displayed
`companies`
    updateCompanies(searchParams);
  }, [searchParams, data]);

  // Function to update query parameters based on user input
  const handleChange = (e) => {
    const newName = e.target.value;
    setSearchParams({ name: newName });
  };

  return (
    <>
      <input
```

```
        type="text"
        value={searchParams.get("name") || ""}
        onChange={handleChange}
      />
      <ul className="companyList">
        {companies.map(({ id, companyName, companyDescription }) => (
          <li key={id}>
            <h2>{companyName}</h2>
            <p>{companyDescription}</p>
          </li>
        ))}
      </ul>
    </>
  );
};
```

Вот подробное описание кода:

1. Мы импортируем необходимые зависимости: useState, useEffect, и useSearchParams из React Router;
2. Мы инициализируем состояние внутри компонента с помощью метода useState. Состояние companies содержит данные для отображения, и мы используем setCompanies функцию для его обновления;
3. Мы создаём функцию updateCompanies для фильтрации данных на основе параметров запроса. Функция принимает текущие параметры запроса в качестве URLSearchParams экземпляра и использует их для фильтрации data. В данном случае она фильтрует по параметру "name";
4. Мы используем useEffect хук для запуска updateCompanies функции при монтировании компонента и всякий раз, когда изменяются параметры запроса. Это гарантирует синхронизацию отображаемых данных с URL-адресом;
5. Функция handleChange обновляет параметр запроса "name" в ответ на ввод пользователя. При изменении значения поля ввода она вызывает функцию setSearchParams с новым значением параметра;
6. В операторе return компонента мы отображаем поле ввода для ввода данных пользователем и список компаний на основе отфильтрованных данных.

Этот код демонстрирует, как интегрировать useSearchParams и React Router для создания динамического веб-приложения, реагирующего на изменения параметров запроса.

6. Вопросы:

1) Что такое **query parameters (параметры запроса)** в URL?
2) Где в URL располагаются query parameters?
3) Что означают key и value в параметрах запроса?
4) Приведите пример URL с двумя query parameters.

5) Чем параметры запроса отличаются от параметров маршрута (path parameters)?
6) Для чего используется хук useSearchParams?
7) Что возвращает useSearchParams()?
8) Что такое searchParams?
9) Что такое setSearchParams?
10) Для чего используется следующий код?