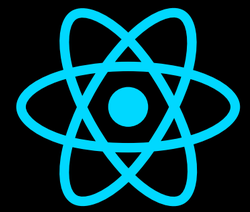
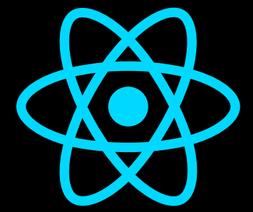


Rotas no React

NATHALIA MACEDO
DESENVOLVEDORA FULL STACK

O que iremos aprender?



- O que é rotas no React?
- Instalação?
- Como definir as rotas?
- BrowserRouter
- Routes - path, element, path com valor ‘*’, rota como index
- Link e Outlet
- Mão no código!
- useNavigate - Naveg1ação progmática,redirecionamento condicional
- Diferença entre Link e useNavigate
- useLocation - pathname,search,hash,state

O que é rotas no React?

Em React, rotas são usadas para criar aplicativos de página única (SPA - Single Page Applications), onde diferentes componentes são renderizados com base na URL atual do navegador. Para gerenciar rotas em React, você pode usar a biblioteca react-router-dom, que é a mais comum para esse propósito. Aqui está uma visão geral do processo:

Instalação

Primeiro, você precisa instalar o react-router-dom no seu projeto. Você pode fazer isso executando o seguinte comando no terminal:

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

Onde definir as rotas?

Definimos as rotas dentro de App.js porque geralmente é o ponto de entrada principal do aplicativo React. É uma prática comum concentrar a lógica de roteamento e a estrutura de navegação principal nesse componente, pois isso facilita a manutenção e organização do código.

BrowserRouter

O BrowserRouter é um tipo de componente fornecido pela biblioteca react-router-dom que fornece a funcionalidade de roteamento para um aplicativo React.

O BrowserRouter utiliza a API de histórico do HTML5 (history) para manter a sincronização entre a URL exibida no navegador e o estado do seu aplicativo React. Isso significa que ele permite que você crie um aplicativo de página única (SPA) que muda dinamicamente os componentes renderizados com base na URL atual do navegador, sem recarregar a página inteira.

Para usar o BrowserRouter, você precisa envolver sua aplicação com ele no nível mais alto possível, geralmente em torno de todo o seu aplicativo, como mostrado no exemplo a seguir:

BrowserRouter

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';
import App from './App';

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

BrowserRouter

Neste exemplo, o componente App e todos os seus componentes filhos terão acesso às funcionalidades de roteamento fornecidas pelo BrowserRouter.

Em resumo, o BrowserRouter é um componente essencial fornecido pelo react-router-dom que permite a criação de aplicativos React com roteamento baseado em URL. Ele gerencia a sincronização entre a URL do navegador e o estado do aplicativo, permitindo a criação de aplicativos de página única (SPAs) que respondem dinamicamente às mudanças na URL.

Routes

O <Routes> é um componente que geralmente é usado para encapsular e definir as rotas do aplicativo. Ele é responsável por agrupar e organizar as rotas do aplicativo. Dentro do componente <Routes>, você define todas as suas rotas usando os componentes <Route> fornecidos pelo react-router-dom.

Routes

```jsx
// App.js

import React from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';

const App = () => {
  return (
    <BrowserRouter>
      <Routes>
        {/* configurar as rotas aqui dentro...*/}
      </Routes>
    </BrowserRouter>
  );
};

export default App;
```

Route

O <Route> é um componente usado para definir uma rota em um aplicativo React. Ele mapeia um URL para um componente específico que será renderizado quando a URL corresponder ao caminho especificado.

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

path

O atributo path em um <Route> é usado para especificar o caminho da rota. É o padrão de URL que deve corresponder para que o componente associado seja renderizado.

element

O atributo element em um <Route> recebe o componente que será renderizado naquela rota específica quando o caminho da URL corresponder ao path especificado.

path com valor ‘ * ‘

Quando o path de uma rota é definido como "*", essa rota captura qualquer URL que não corresponda às rotas já especificadas. É útil para criar páginas de erro, como uma página de erro 404, que é exibida quando um usuário tenta acessar uma URL que não existe.

path com valor ‘ * ‘

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

Rota como index

Quando você define uma rota como index em um aplicativo React com roteamento, significa que esta rota será a rota padrão, ou seja, a rota que será renderizada quando nenhuma outra rota corresponder à URL atual do navegador.

Rota como index

Isso é especialmente útil para definir a página inicial ou raiz do seu aplicativo. Por exemplo, se você tem um aplicativo com várias páginas, como Home, Sobre, Contato, etc., a rota index será a primeira a ser carregada quando o usuário acessar o aplicativo.

Rota como index

```
/*Neste exemplo, a rota com path="/",
que é a rota raiz, é definida como index.
Isso significa que quando o usuário acessar
o aplicativo, por exemplo, em http://meuapp.com/,
a página inicial (<Home />) será renderizada
automaticamente, porque a rota raiz
é a rota padrão.*/

<Route path="/" element={<Home />} />
<Route path="/about" element={<About />} />
<Route path="/contact" element={<Contact />} />
<Route path="*" element={<NotFound />} />
```

Rota como index

Neste exemplo, a rota com path="/", que é a rota raiz, é definida como index. Isso significa que quando o usuário acessar o aplicativo, por exemplo, em http://meuapp.com/, a página inicial (<Home />) será renderizada automaticamente, porque a rota raiz é a rota padrão.

Se o usuário acessar uma URL que não corresponda a nenhuma das rotas definidas, como http://meuapp.com/qualquercoisa, a rota path="*" será correspondida, e a página de erro (<NotFound />) será renderizada.

Rota como index

Essa é uma maneira simples e eficaz de garantir que seu aplicativo tenha uma página inicial definida e que possa lidar com URLs não correspondidas de forma apropriada.

Link

O <Link> é um componente usado para criar links de navegação entre diferentes rotas em um aplicativo React. Em vez de usar a tag <a> com href, você usa <Link> para navegar entre as páginas no aplicativo sem recarregar a página inteira.

Link

O <Link> é um componente usado para criar links de navegação entre diferentes rotas em um aplicativo React. Em vez de usar a tag <a> com href, você usa <Link> para navegar entre as páginas no aplicativo sem recarregar a página inteira.

```
<Link to="/about">Sobre nós</Link>
```

Neste exemplo, ao clicar no link "Sobre nós", o usuário será direcionado para a rota "/about" sem recarregar a página. Isso mantém o estado do aplicativo e fornece uma experiência de navegação mais suave.

Outlet

O <Outlet> é um componente muito útil quando você está trabalhando com layouts em um aplicativo React usando o react-router-dom. Ele funciona como um "ponto de saída" onde o conteúdo da rota atual será renderizado. Isso é particularmente útil quando você tem um layout padrão que deseja manter consistente em todas as páginas do seu aplicativo.

Imagine que você tenha um layout comum para todas as páginas do seu aplicativo, algo assim:

Outlet

O <Outlet> é um componente muito útil quando você está trabalhando com layouts em um aplicativo React usando o react-router-dom. Ele funciona como um "ponto de saída" onde o conteúdo da rota atual será renderizado. Isso é particularmente útil quando você tem um layout padrão que deseja manter consistente em todas as páginas do seu aplicativo.

Imagine que você tenha um layout comum para todas as páginas do seu aplicativo, algo assim:

Outlet

```
function Layout() {
    return (
      <div>
        <Header />
        <Sidebar />
        <main>
          {/* Aqui é onde você quer que o
           conteúdo da rota atual seja renderizado */}
          {/* Este é o lugar onde você usará o Outlet */}
          <Outlet />
        </main>
        <Footer />
      </div>
    );
  }
```

Outlet

Aqui, o <Outlet /> é onde o conteúdo da rota atual será renderizado. Por exemplo, se a rota atual for "/about", o componente associado a essa rota será renderizado dentro do <Outlet />. Se a rota atual for "/contact", o componente associado a essa rota será renderizado no mesmo lugar.

Isso permite que você mantenha um layout consistente em todo o seu aplicativo, independentemente da rota atual. Você pode ter um cabeçalho, um menu lateral, um rodapé e outras partes do layout que permanecem constantes, enquanto o conteúdo da página pode mudar com base na rota atual.

Outlet

Portanto, o <Outlet> é um componente fundamental quando se trata de criar layouts flexíveis e reutilizáveis em aplicativos React com roteamento. Ele permite que você mantenha um design consistente em todo o seu aplicativo, independentemente da rota atual.

Mão no código!

Crie três páginas: Home, Contato, e sobre mim! Na home, coloque um Link, para levar para as outras duas. Dentro de cada uma das páginas, é necessário ter um botão, que leve de volta para a Home!

useNavigate

useNavigate é um hook fornecido pela biblioteca react-router-dom que permite a navegação programática entre páginas em um aplicativo React. Ao contrário do componente <Link>, que é usado para criar links de navegação na interface do usuário, useNavigate é usado dentro de funções de eventos, como manipuladores de eventos de clique ou funções assíncronas, para realizar a navegação.

Uso Básico

Para usar o useNavigate, primeiro você precisa importá-lo do react-router-dom:

useNavigate

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

Em seguida, você pode chamar o useNavigate dentro do seu componente funcional:

useNavigate

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

    const handleClick = () => {
      // Navega para a rota "/about" quando o botão é clicado
      navigate('/about');
    };

    return (
      <div>
        <button onClick={handleClick}>Ir para a página Sobre</button>
      </div>
    );
  };
```

Navegação Programática

useNavigate permite que você navegue programaticamente para uma rota específica em resposta a eventos ou condições de negócio dentro do seu aplicativo. Isso é útil quando você precisa navegar após realizar alguma ação ou quando a navegação é baseada em lógica dinâmica.

Navegação Programática

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

    const handleLogin = async () => {
      // Faz login do usuário...

      // Se o login for bem-sucedido, navega para a página inicial
      navigate('/');
    };

    return (
      <div>
        <button onClick={handleLogin}>Login</button>
      </div>
    );
  };
```

Redirecionamento Condicional

Você também pode usar o useNavigate para redirecionamento condicional com base em alguma lógica ou estado do componente.

```
const MyComponent = ({ isLoggedIn }) => {
    const navigate = useNavigate();

    // Se o usuário estiver logado,
    //redirecione para a página de perfil,
    // senão, para a página de login
    if (isLoggedIn) {
      navigate('/profile');
    } else {
      navigate('/login');
    }

    return (
      <div>
        {/* Conteúdo do componente */}
      </div>
    );
  };
```

Diferença entre Link e useNavigate

Em resumo, <Link> é para navegação via interface do usuário, enquanto useNavigate é para navegação programática em resposta a eventos ou lógica dentro de componentes funcionais. Ambos são importantes e são usados em diferentes contextos, dependendo das necessidades do aplicativo.

useLocation

O useLocation é um hook fornecido pelo react-router-dom que permite acessar o objeto de localização atual do navegador. Esse objeto contém informações sobre a URL atual, como o caminho, a pesquisa (query string), o hash e o estado.

pathname

pathname é uma propriedade do objeto de localização que representa o caminho da URL atual, excluindo a busca (query string) e o hash. É a parte da URL que vem após o nome do host e a porta, até o início da busca ou do hash.

search

search é uma propriedade do objeto de localização que representa a parte da URL após o ?, contendo a query string. A query string geralmente é usada para enviar dados para o servidor ou para controlar o comportamento da página.

hash

hash é uma propriedade do objeto de localização que representa a parte da URL após o #. O hash é frequentemente usado para navegação na mesma página (âncoras) ou para fornecer links diretos para seções específicas de uma página.

state

state é uma propriedade do objeto de localização que permite passar dados entre rotas no react-router-dom. Ele fornece uma maneira de passar informações adicionais para a próxima rota, sem a necessidade de usar a query string ou o hash da URL.

Suponha que você tenha uma rota que leva o usuário a uma página de detalhes de um produto. Você pode querer passar informações específicas sobre o produto, como o ID do produto, o nome, a descrição, etc., para a próxima rota. Em vez de incluir todas essas informações na URL como parâmetros de consulta (query string) ou como parte do hash, você pode usar o state para passar esses dados de forma mais conveniente.

state

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

function ProductList() {
  return (
    <ul>
      <li><Link to={{ pathname: '/product/1', state: { id: 1, name: 'Product 1' } }}>Product 1</Link></li>
      <li><Link to={{ pathname: '/product/2', state: { id: 2, name: 'Product 2' } }}>Product 2</Link></li>
    </ul>
  );
}

function ProductDetail() {
  const location = useLocation();
  const { id, name } = location.state; // Acessando os dados do state

  return (
    <div>
      <h2>Product Detail</h2>
      <p>ID: {id}</p>
      <p>Name: {name}</p>
    </div>
  );
}
```

Mão no código!

Você está desenvolvendo um aplicativo de catálogo de filmes. Neste aplicativo, você tem duas páginas: uma página inicial que exibe todos os filmes disponíveis e uma página de detalhes que exibe informações específicas de um filme selecionado. Além disso, você deseja fornecer um recurso de pesquisa para permitir que os usuários pesquisem filmes por título.

1. Implemente um aplicativo React usando o create-react-app.
2. Configure rotas usando o react-router-dom para as duas páginas: a página inicial (/) e a página de detalhes (/movie/:id).
3. Na página inicial, exiba uma lista de todos os filmes disponíveis. Cada filme na lista deve ser um link para a página de detalhes correspondente.

Mão no código!

1. Na página de detalhes, exiba informações específicas do filme, como título, descrição, ano de lançamento, etc.
2. Implemente um campo de pesquisa na página inicial que permita aos usuários pesquisar filmes por título.
3. Ao clicar em um filme na lista de resultados da pesquisa, os usuários devem ser redirecionados para a página de detalhes correspondente.
4. Use os hooks useState, useLocation e useNavigate conforme necessário para gerenciar o estado da pesquisa e para navegação entre páginas.