Hooks Adicionais

Hooks que não são nativos do react, que fazem parte de uma biblioteca como React Router, Redux, etc.

Hooks são lógicas, enquanto componentes são visuais.

React Router

React Router é uma biblioteca de roteamento para aplicações React, permitindo navegação entre diferentes páginas sem recarregar o navegador.

Hooks são usados dentro de funções/componentes React para acessar o estado da rota ou controlar o fluxo de navegação.

Componentes são usados no JSX para declarar as rotas e construir a UI de navegação.

Instalar: `npm install react-router-dom`

Hooks Principais

Importar: `import { useNavigate, useParams, useLocation, useMatch, useSearchParams, useOutlet, useRoutes } from “react-router-dom”`

useNavigate: Navega entre rotas.

useParams: Retorna um objeto com os parâmetros da URL atual.

useLocation: Retorna o objeto de localização atual (path, search, hash).

useMatch: Verifica se a URL atual bate com um caminho.

useSearchParams: Lê e seta (define) parâmetros de busca (query strings).

useOutlet: Renderiza conteúdo filho em rotas aninhadas.

useRoutes: Define rotas como objetos em vez de JSX.

Componentes Principais

Importar: `import { BrowserRouter, Routes, Route, Link, NavLink, Navigate, Outlet, Form, ScrollRestoration } from “react-router-dom”`

BrowserRouter: Envolve toda a aplicação para ativas o roteamento.

Routes: Agrupa várias <Route />

Route: Define uma rota especifica com path e element.

Link: Link de navegação, como um <a> mas sem recarregar a pagina.

NavLink: Igual ao Link, mas com estilos ativos automáticos.

Navigate: Redirecionamento programático dentro do JSX.

Outlet: Ponto de saída para rotas aninhadas (filhas).

Form: Componente que integra com actions e loaders.

ScrollRestoration: Restaura a posição de scroll entre navegação.

Utilização / Exemplo Principal:

App.jsx

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Home from './pages/Home';
import Sobre from './pages/Sobre';
import Contato from './pages/Contato';

function App() {
  return (
    <BrowserRouter>
```

```
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/sobre" element={<Sobre />} />
        <Route path="/contato" element={<Contato />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

Navbar.jsx

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

function Navbar() {
  return (
    <nav>
      <Link to="/">Início</Link>
      <Link to="/sobre">Sobre</Link>
      <Link to="/contato">Contato</Link>
    </nav>
  );
}
```

Redirect -

```
import { Navigate } from 'react-router-dom';
<Route path="/antigo" element={<Navigate to="/" />} />
```

Rota com Parametro -

```
<Route path="/usuario/:id" element={<PerfilUsuario />} />

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

```
function PerfilUsuario() {
  const { id } = useParams();
  return <p>ID do usuário: {id}</p>;
}
```

404 -

```
<Route path="*" element={<h1>Página não encontrada</h1>} />
```

React Query (TanStack Query)

React Query (TanStack Query) é uma biblioteca para gerenciamento de estado assíncrono no React, usada principalmente para buscar, armazenar em cache, sincronizar e atualizar dados de servidores (como APIs REST ou GraphQL).

Instalar: `npm install @tanstack/react-query`

Hooks Principais

Importar: `import { useQuery, useMutation, useQueryClient, useInfiniteQuery, useIsFetching, useIsMutating } from “@tanstack/react-query ”`

useQuery: Usado para buscar dados (GET). Executa automaticamente quando o componente é montado:

```
const { data, isLoading, error } = useQuery({
  queryKey: ['dados'],
  queryFn: fetchFn,
})
```

useMutation: Usado para enviar dados (POST, PUT, DELETE). Você chama manualmente com mutate:

```
const mutation = useMutation({ mutationFn: enviarFn })
mutation.mutate({ nome: 'João' })
```

useQueryClient: Acesso direto ao QueryClient, útil para invalidar cache, fazer refetch, atualizar dados manualmente:

```
const queryClient = useQueryClient()
queryClient.invalidateQueries({ queryKey: ['dados'] })
```

useInfiniteQuery: Para paginação infinita (ex: scroll infinito). Ele lida com várias "páginas" de dados:

```
const { data, fetchNextPage, hasNextPage } = useInfiniteQuery({
  queryKey: ['posts'],
  queryFn: fetchFn,
  getNextPageParam: (lastPage) => lastPage.nextCursor,
});
```

useIsFetching, useIsMutating: Usados para detectar se há requisições em andamento:

```
const isFetching = useIsFetching()  // true/false se está buscando
const isMutating = useIsMutating()  // true/false se está mutando
```

React Hook Form

O React Hook Form é uma biblioteca leve para gerenciamento de formulários no React, conhecida por sua performance e simplicidade.

Instalar: `npm install react-hook-form`

Importar: `import { useForm, useFormContext, useFieldArray, useWatch, useController } from 'react-hook-form'`

useForm: Cria e gerencia formulários com validação e estado.

useFormContext: Compartilha o estado do formulário entre componentes aninhados.

useFieldArray: Gerencia campos de formulário dinâmicos (ex: listas de inputs).

useWatch: Observa mudanças em campos específicos do formulário.

useController: Integra componentes controlados com o React Hook Form.

Framer Motion

O Framer Motion é uma biblioteca de animações para React que facilita muito a criação de animações suaves e interações modernas em interfaces.

Instalar: `npm install framer-motion`

Hooks

Importar: `import { useAnimation, useCycle, useInView, useMotionValue } from "framer-motion"`

useAnimation: Controla animações programaticamente.

useCycle: Alterna entre diferentes estados ou estilos animados.

useInView: Detecta quando um elemento entra na viewport.

useMotionValue: Cria valores reativos usados em animações.

Componentes

Importar: `import { motion, AnimatePresence, LazyMotion, domAnimation, LayoutGroup, MotionConfig } from 'framer-motion'`

motion: Transforma elementos HTML ou componentes React em elementos animáveis com propriedades como animate, initial, exit, etc.

AnimatePresence: Permite animar elementos que estão sendo removidos do DOM, controlando a animação de saída.

LazyMotion: Otimiza o carregamento da biblioteca, carregando recursos de animação apenas quando necessário.

domAnimation: Conjunto padrão de animações do navegador usado em conjunto com LazyMotion.

LayoutGroup: Agrupa elementos que compartilham animações de layout, garantindo sincronização suave entre eles.

MotionConfig: Define configurações globais de animação, como transition ou reducedMotion.
