Lab 6 React Routes with Parameter

1. Install Dependencies. Make sure you have React and react-router-dom installed in your project.
2. Import Dependencies. Add the following import statements at the top of app.js:

import React from 'react';

import { Outlet, Link } from 'react-router-dom';

1. Define the Component. Create a default functional component called `App.` Outlet is where things will show up:

export default function App() {

return (

<div>

<h1>Bookkeeper</h1>

<nav style={{ borderBottom: 'solid 1px', paddingBottom: '1rem' }}>

<Link to="/invoices">Invoices</Link> |{' '}

<Link to="/expenses">Expenses</Link>

</nav>

<Outlet />

</div>

);

}

1. Export the Component. At the end of your file, add the following export statement:

export default App;

1. Create an array called `invoices` with the invoice data. Each invoice should have the following properties: `name`, `number`, `amount`, and `due`.

let invoices = [

{

name: "Santa Monica",

number: 1995,

amount: "$10,800",

due: "12/05/1995"

},

{

name: "Stankonia",

number: 2000,

amount: "$8,000",

due: "10/31/2000"

},

{

name: "Ocean Avenue",

number: 2003,

amount: "$9,500",

due: "07/22/2003"

},

{

name: "Tubthumper",

number: 1997,

amount: "$14,000",

due: "09/01/1997"

},

{

name: "Wide Open Spaces",

number: 1998,

amount: "$4,600",

due: "01/27/2998"

}

];

1. Retrieve all invoices. Create a function called `getInvoices` that returns the `invoices` array.

export function getInvoices() {

return invoices;

}

1. Retrieve a specific invoice. Create a function called `getInvoice` that takes a `number` parameter and returns the invoice with a matching `number`.

export function getInvoice(number) {

return invoices.find(invoice => invoice.number === number);

}

1. Delete an invoice. Create a function called `deleteInvoice` that takes a `number` parameter and removes the invoice with a matching `number` from the `invoices` array.

export function deleteInvoice(number) {

invoices = invoices.filter(invoice => invoice.number !== number);

}

1. Create three files: expenses, invoices, and invoice, inside a folder called routes.
2. Now for the index.js file. Import dependencies. Make sure the file has:

import React from 'react';

import ReactDOM from 'react-dom';

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

import App from './App';

import Expenses from './routes/expenses';

import Invoices from './routes/invoices';

import Invoice from './routes/invoice';

import './index.css';

1. Define the Routes and components. Use the `ReactDOM.createRoot` function to create a root instance for rendering the application.
   1. Wrap the application with a `BrowserRouter` component to enable routing functionality.
   2. Use the `Routes` component to define the routes of your application.
   3. Use the `Route` component to specify the path and corresponding component for each route.
   4. Nest routes by using nested `Route` components.
   5. Provide the `element` prop with the desired component to render for each route. You can use inline JSX code within the `element` prop to define the UI for each route.

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<BrowserRouter>

<Routes>

<Route path="/" element={<App />}>

<Route path="expenses" element={<Expenses />} />

<Route path="invoices" element={<Invoices />}>

<Route

index

element={

<main style={{ padding: '1rem' }}>

<p>Select an invoice</p>

</main>

}

/>

<Route path=":invoiceId" element={<Invoice />} />

</Route>

<Route

path="\*"

element={

<main style={{ padding: '1rem' }}>

<p>There's nothing here!</p>

</main>

}

/>

</Route>

</Routes>

</BrowserRouter>

);

1. In routes/expenses.js, create a detailed Expenses component:

import \* as React from 'react';

export default function Expenses() {

return (

<main style={{ padding: '1rem 0' }}>

<h2>Expenses</h2>

</main>

);

}

1. Export the component.
2. For invoices.js, import dependencies. Add the following import statements at the top of your file:

import React from 'react';

import {

useLocation,

NavLink,

Outlet,

useSearchParams,

} from 'react-router-dom';

import { getInvoices } from '../data';

1. Define the QueryNavLink Component:
   1. Create a functional component called `QueryNavLink` that wraps the `NavLink` component.
   2. Use the `useLocation` hook to get the current location.
   3. Pass the updated location search string to the `NavLink` component.

function QueryNavLink({ to, ...props }) {

let location = useLocation();

return <NavLink to={to + location.search} {...props} />;

}

1. Define the invoices component:
   1. Create a default functional component called `Invoices`.
   2. Retrieve the invoices using the `getInvoices` function from the '../data' module.
   3. Use the `useSearchParams` hook to get and update the search parameters.
   4. Return a JSX structure that renders the invoices list and provides a search filter input field.

export default function Invoices() {

let invoices = getInvoices();

let [searchParams, setSearchParams] = useSearchParams({ replace: true });

return (

<div style={{ display: 'flex' }}>

<nav style={{ borderRight: 'solid 1px', padding: '1rem' }}>

<input

value={searchParams.get('filter') || ''}

onChange={(event) => {

let filter = event.target.value;

if (filter) {

setSearchParams({ filter }, { replace: true });

} else {

setSearchParams({}, { replace: true });

}

}}

/>

{invoices

.filter((invoice) => {

let filter = searchParams.get('filter');

if (!filter) return true;

let name = invoice.name.toLowerCase();

return name.startsWith(filter.toLowerCase());

})

.map((invoice) => (

<QueryNavLink

key={invoice.number}

style={({ isActive }) => {

return {

display: 'block',

margin: '1rem 0',

color: isActive ? 'red' : '',

};

}}

to={`/invoices/${invoice.number}`}

>

{invoice.name}

</QueryNavLink>

))}

</nav>

<Outlet />

</div>

);

}

1. Export the component. At the end of your file, add the following export statement:

export default Invoices;

1. Note: The `Invoices` component is a functional component that renders a navigation bar with a search filter input field and a list of invoices. It uses the `QueryNavLink` component to handle navigation with search parameters. The list of invoices is filtered based on the search filter input.
2. Now we will work on invoice.js. Import dependencies. Add the following import statements at the top of your file:

import React from 'react';

import {

useLocation,

NavLink,

Outlet,

useSearchParams,

} from 'react-router-dom';

import { getInvoices } from '../data';

1. Define the QueryNavLink Component:
   1. Create a functional component called `QueryNavLink` that wraps the `NavLink` component.
   2. Use the `useLocation` hook from react-router-dom to get the current location.
   3. Pass the updated location search string to the `NavLink` component.

function QueryNavLink({ to, ...props }) {

let location = useLocation();

return <NavLink to={to + location.search} {...props} />;

}

1. Define the Invoices Component:
   1. Create a default functional component called `Invoices`.
   2. Retrieve the invoices using the `getInvoices` function from the '../data' module.
   3. Use the `useSearchParams` hook from react-router-dom to get and update the search parameters.
   4. Return a JSX structure that renders the invoices list and provides a search filter input field.

export default function Invoices() {

let invoices = getInvoices();

let [searchParams, setSearchParams] = useSearchParams({ replace: true });

return (

<div style={{ display: 'flex' }}>

<nav style={{ borderRight: 'solid 1px', padding: '1rem' }}>

<input

value={searchParams.get('filter') || ''}

onChange={(event) => {

let filter = event.target.value;

if (filter) {

setSearchParams({ filter }, { replace: true });

} else {

setSearchParams({}, { replace: true });

}

}}

/>

{invoices

.filter((invoice) => {

let filter = searchParams.get('filter');

if (!filter) return true;

let name = invoice.name.toLowerCase();

return name.startsWith(filter.toLowerCase());

})

.map((invoice) => (

<QueryNavLink

key={invoice.number}

style={({ isActive }) => {

return {

display: 'block',

margin: '1rem 0',

color: isActive ? 'red' : '',

};

}}

to={`/invoices/${invoice.number}`}

>

{invoice.name}

</QueryNavLink>

))}

</nav>

<Outlet />

</div>

);

}

1. Export the component.
2. Note: The `Invoices` component is a functional component that renders a navigation bar with a search filter input field and a list of invoices. It uses the `QueryNavLink` component to handle navigation with search parameters. The list of invoices is filtered based on the search filter input.