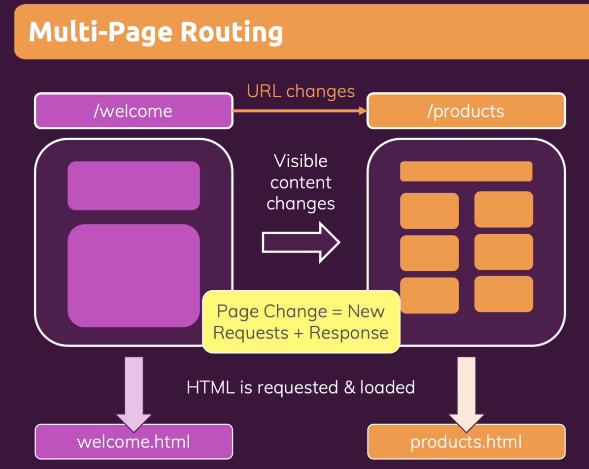
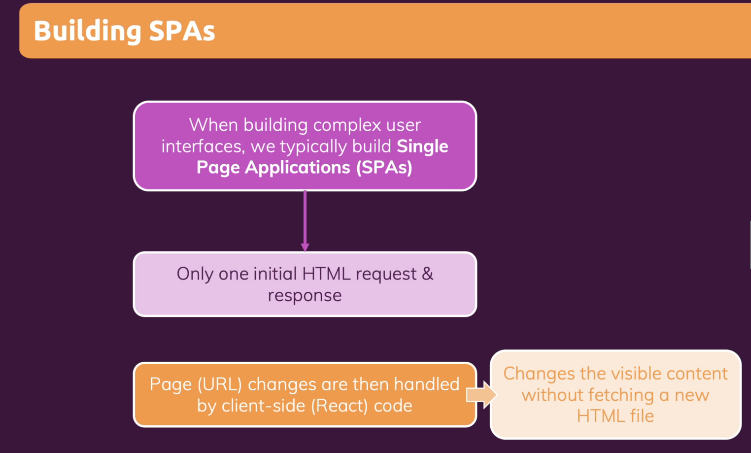
Section 20: Building a Multi-Page SPA with React Router

266. What is Routing & Why



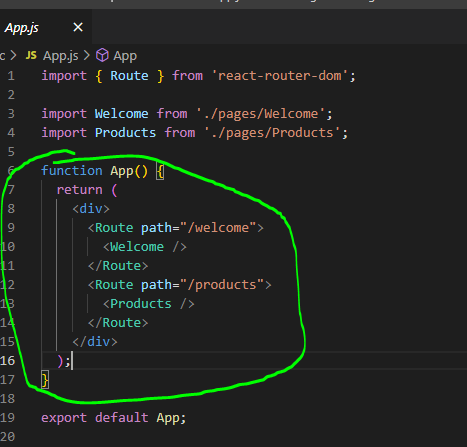


267. Installing React Router

Step 1: Clone source code <https://github.com/academind/react-complete-guide-code/tree/20-building-mpas-with-react-router>

Step 2: run npm install react-router-dom@5

268. Defining & Using Routes



Step 1: add new folder pages + Welcome.js, Products.js

In Welcomes.js

const Welcome = () => {

  return <h1>The Welcome Page</h1>;

};

export default Welcome;

in Product.js

const Products = () => {

  return <h1>The Products Page</h1>;

};

export default Products;

Step 2: in App.js

import { Route } from 'react-router-dom';

import Welcome from './pages/Welcome';

import Products from './pages/Products';

function App() {

  return (

    <div>

      <Route path="/welcome">

        <Welcome />

      </Route>

      <Route path="/products">

        <Products />

      </Route>

    </div>

  );

}

export default App;

// our-domain.com/welcome => Welcome Component

// our-domain.com/products => Products Component

Step 3: in index.js

import ReactDOM from 'react-dom/client';

import { BrowserRouter } from 'react-router-dom';

import './index.css';

import App from './App';

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

root.render(

  <BrowserRouter>

    <App />

  </BrowserRouter>

);

Step 4: test result

269. Working with Links

Step 1: create folder components + MainHeader.js

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

import classes from './MainHeader.module.css';

const MainHeader = () => {

  return (

    <header className={classes.header}>

      <nav>

        <ul>

          <li>

            <Link to='/welcome'>Welcome</Link>

          </li>

          <li>

            <Link to='/products'>Products</Link>

          </li>

        </ul>

      </nav>

    </header>

  );

};

export default MainHeader;

Step 2: add css MainHeader.module.css

.header {

  width: 100%;

  height: 5rem;

  background-color: #044599;

  padding: 0 10%;

}

.header nav {

  height: 100%;

}

.header ul {

  height: 100%;

  list-style: none;

  display: flex;

  padding: 0;

  margin: 0;

  align-items: center;

  justify-content: center;

}

.header li {

  margin: 0 1rem;

  width: 5rem;

}

.header a {

  color: white;

  text-decoration: none;

}

.header a:hover,

.header a:active {

  color: #95bcf0;

  padding-bottom: 0.25rem;

  border-bottom: 4px solid #95bcf0;

}

270. Using NavLinks

Hover when click to that page

Step 1 In MainHeader.js

import { NavLink } from 'react-router-dom';

import classes from './MainHeader.module.css';

const MainHeader = () => {

  return (

    <header className={classes.header}>

      <nav>

        <ul>

          <li>

            <NavLink activeClassName={classes.active} to='/welcome'>

              Welcome

            </NavLink>

          </li>

          <li>

            <NavLink activeClassName={classes.active} to='/products'>

              Products

            </NavLink>

          </li>

        </ul>

      </nav>

    </header>

  );

};

export default MainHeader;

Step 2 in MainHeader.module.css

.header a:hover,

.header a:active,

.header a.active {

  color: #95bcf0;

  padding-bottom: 0.25rem;

  border-bottom: 4px solid #95bcf0;

}

271. Adding Dynamic Routes with Params + Extracting Route Params

Step 1 add new pages/PrudctDetail.js

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

const ProductDetail = () => {

  const params = useParams();

  console.log(params.productId);

  return (

    <section>

      <h1>Product Detail</h1>

      <p>{params.productId}</p>

    </section>

  );

};

export default ProductDetail;

Step 2 in App.js

import { Route } from 'react-router-dom';

import Welcome from './pages/Welcome';

import Products from './pages/Products';

import ProductDetail from './pages/ProductDetail';

import MainHeader from './components/MainHeader';

function App() {

  return (

    <div>

      <MainHeader />

      <main>

        <Route path='/welcome'>

          <Welcome />

        </Route>

        <Route path='/products'>

          <Products />

        </Route>

        <Route path='/product-detail/:productId'>

          <ProductDetail />

        </Route>

      </main>

    </div>

  );

}

export default App;

// our-domain.com/welcome => Welcome Component

// our-domain.com/products => Products Component

// our-domain.com/product-detail/a-book

273. Using “Switch” and “exact” For Configuring Routes (make sure only one link work)

Step 1 in Products.js

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

const Products = () => {

  return (

    <section>

      <h1>The Products Page</h1>

      <ul>

        <li>

          <Link to='/products/p1'>A Book</Link>

        </li>

        <li>

          <Link to='/products/p2'>A Carpet</Link>

        </li>

        <li>

          <Link to='/products/p3'>An Online Course</Link>

        </li>

      </ul>

    </section>

  );

};

export default Products;

Step 2 in App.js (exact only allow show when match)

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

import Welcome from './pages/Welcome';

import Products from './pages/Products';

import ProductDetail from './pages/ProductDetail';

import MainHeader from './components/MainHeader';

function App() {

  return (

    <div>

      <MainHeader />

      <main>

        <Switch>

          <Route path='/welcome'>

            <Welcome />

          </Route>

          <Route path='/products' exact>

            <Products />

          </Route>

          <Route path='/products/:productId'>

            <ProductDetail />

          </Route>

        </Switch>

      </main>

    </div>

  );

}

export default App;

// our-domain.com/welcome => Welcome Component

// our-domain.com/products => Products Component

// our-domain.com/product-detail/a-book

274. Working with Nested Routes

Step 1 in Welcome.js

import { Route } from 'react-router-dom';

const Welcome = () => {

  return (

    <section>

      <h1>The Welcome Page</h1>

      <Route path="/welcome/new-user">

        <p>Welcome, new user!</p>

      </Route>

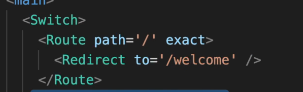
    </section>

  );

};

export default Welcome;

275. Redirecting The User



Step 1 in App.js

import { Route, Switch, Redirect } from 'react-router-dom';

import Welcome from './pages/Welcome';

import Products from './pages/Products';

import ProductDetail from './pages/ProductDetail';

import MainHeader from './components/MainHeader';

function App() {

  return (

    <div>

      <MainHeader />

      <main>

        <Switch>

          <Route path='/' exact>

            <Redirect to='/welcome' />

          </Route>

          <Route path='/welcome'>

            <Welcome />

          </Route>

          <Route path='/products' exact>

            <Products />

          </Route>

          <Route path='/products/:productId'>

            <ProductDetail />

          </Route>

        </Switch>

      </main>

    </div>

  );

}

export default App;

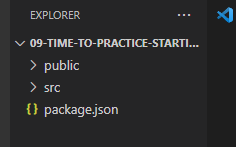
// our-domain.com/welcome => Welcome Component

// our-domain.com/products => Products Component

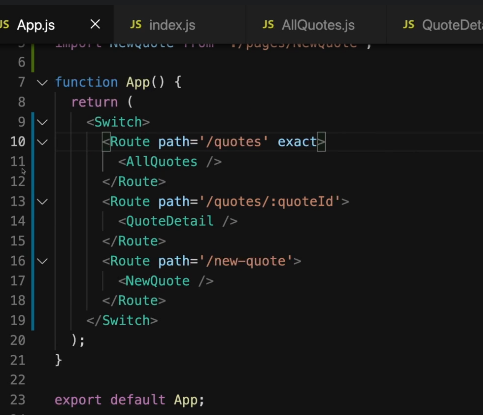
// our-domain.com/product-detail/a-book

276. Onwards to a new Project

Step 1 clone source + npm i + run code



Step 2 in App.js (exact make sure only when match url)



Step 3 in index.js

import ReactDOM from 'react-dom/client';

import { BrowserRouter } from 'react-router-dom';

import './index.css';

import App from './App';

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

root.render(

  <BrowserRouter>

    <App />

  </BrowserRouter>

);

277. Redirecting & Extracting Params

Step 1: in App.js

import { Route, Switch, Redirect } from 'react-router-dom';

import AllQuotes from './pages/AllQuotes';

import QuoteDetail from './pages/QuoteDetail';

import NewQuote from './pages/NewQuote';

function App() {

  return (

    <Switch>

      <Route path='/' exact>

        <Redirect to='/quotes' />

      </Route>

      <Route path='/quotes' exact>

        <AllQuotes />

      </Route>

      <Route path='/quotes/:quoteId'>

        <QuoteDetail />

      </Route>

      <Route path='/new-quote'>

        <NewQuote />

      </Route>

    </Switch>

  );

}

export default App;

Step 2: in QueDetails.js

import { Fragment } from 'react';

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

const QuoteDetail = () => {

  const params = useParams();

  return (

    <Fragment>

      <h1>Quote Detail Page</h1>

      <p>{params.quoteId}</p>

    </Fragment>

  );

};

export default QuoteDetail;

278. Nested Routes

Step 1: in QuoteDetail.js

import { Fragment } from 'react';

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

import Comments from '../components/comments/Comments';

const QuoteDetail = () => {

  const params = useParams();

  return (

    <Fragment>

      <h1>Quote Detail Page</h1>

      <p>{params.quoteId}</p>

      <Route path={`/quotes/${params.quoteId}/comments`}>

        <Comments />

      </Route>

    </Fragment>

  );

};

export default QuoteDetail;

279. Adding a Layout Wrapper Component

Step 1: in folder layout, create new MainNavigation.js

import { NavLink } from 'react-router-dom';

import classes from './MainNavigation.module.css';

const MainNavigation = () => {

  return (

    <header className={classes.header}>

      <div className={classes.logo}>Great Quotes</div>

      <nav className={classes.nav}>

        <ul>

          <li>

            <NavLink to='/quotes' activeClassName={classes.active}>

              All Quotes

            </NavLink>

          </li>

          <li>

            <NavLink to='/new-quote' activeClassName={classes.active}>

              Add a Quote

            </NavLink>

          </li>

        </ul>

      </nav>

    </header>

  );

};

export default MainNavigation;

Step 2: in MainNavigation.module.css

.header {

  width: 100%;

  height: 5rem;

  display: flex;

  padding: 0 10%;

  justify-content: space-between;

  align-items: center;

  background-color: #008080;

}

.logo {

  font-size: 2rem;

  color: white;

}

.nav ul {

  list-style: none;

  display: flex;

  margin: 0;

  padding: 0;

}

.nav li {

  margin-left: 1.5rem;

  font-size: 1.25rem;

}

.nav a {

  text-decoration: none;

  color: #88dfdf;

}

.nav a:hover,

.nav a:active,

.nav a.active {

  color: #e6fcfc;

}

Step 3: in Layout.js

import { Fragment } from 'react';

import classes from './Layout.module.css';

import MainNavigation from './MainNavigation';

const Layout = (props) => {

  return (

    <Fragment>

      <MainNavigation />

      <main className={classes.main}>{props.children}</main>

    </Fragment>

  );

};

export default Layout;

Step 4: then go to App.js

import { Route, Switch, Redirect } from 'react-router-dom';

import AllQuotes from './pages/AllQuotes';

import QuoteDetail from './pages/QuoteDetail';

import NewQuote from './pages/NewQuote';

import Layout from './components/layout/Layout';

function App() {

  return (

    <Layout>

      <Switch>

        <Route path='/' exact>

          <Redirect to='/quotes' />

        </Route>

        <Route path='/quotes' exact>

          <AllQuotes />

        </Route>

        <Route path='/quotes/:quoteId'>

          <QuoteDetail />

        </Route>

        <Route path='/new-quote'>

          <NewQuote />

        </Route>

      </Switch>

    </Layout>

  );

}

export default App;

280. Adding Dummy Data & More Content

Step 1: in AllQuotes.js (note: QuoteList had code before)

import QuoteList from '../components/quotes/QuoteList';

const DUMMY\_QUOTES = [

  { id: 'q1', author: 'Max', text: 'Learning React is fun!' },

  { id: 'q2', author: 'Maximilian', text: 'Learning React is great!' },

];

const AllQuotes = () => {

  return <QuoteList quotes={DUMMY\_QUOTES} />

};

export default AllQuotes;

Step 2: in NewQuote.js

import QuoteForm from '../components/quotes/QuoteForm';

const NewQuote = () => {

  const addQuoteHandler = (quoteData) => {

    console.log(quoteData);

  };

  return <QuoteForm onAddQuote={addQuoteHandler} />;

};

export default NewQuote;

281. Outputting Data on the “Details” Page

Step 1 in QuoteItem.js

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

import classes from './QuoteItem.module.css';

const QuoteItem = (props) => {

  return (

    <li className={classes.item}>

      <figure>

        <blockquote>

          <p>{props.text}</p>

        </blockquote>

        <figcaption>{props.author}</figcaption>

      </figure>

      <Link className='btn' to={`/quotes/${props.id}`}>

        View Fullscreen

      </Link>

    </li>

  );

};

export default QuoteItem;

Step 2 in QuoteDetail.js

import { Fragment } from 'react';

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

import HighlightedQuote from '../components/quotes/HighlightedQuote';

import Comments from '../components/comments/Comments';

const DUMMY\_QUOTES = [

  { id: 'q1', author: 'Max', text: 'Learning React is fun!' },

  { id: 'q2', author: 'Maximilian', text: 'Learning React is great!' },

];

const QuoteDetail = () => {

  const params = useParams();

  const quote = DUMMY\_QUOTES.find((quote) => quote.id === params.quoteId);

  if (!quote) {

    return <p>No quote found!</p>;

  }

  return (

    <Fragment>

      <HighlightedQuote text={quote.text} author={quote.author} />

      <Route path={`/quotes/${params.quoteId}/comments`}>

        <Comments />

      </Route>

    </Fragment>

  );

};

export default QuoteDetail;

282. Adding a “Not Found” Page

Step 1 create new NotFound.js in pages

const NotFound = () => {

  return (

    <div className='centered'>

      <p>Page not found!</p>

    </div>

  );

};

export default NotFound;

Step 2 in App.js

import { Route, Switch, Redirect } from 'react-router-dom';

import AllQuotes from './pages/AllQuotes';

import QuoteDetail from './pages/QuoteDetail';

import NewQuote from './pages/NewQuote';

import NotFound from './pages/NotFound';

import Layout from './components/layout/Layout';

function App() {

  return (

    <Layout>

      <Switch>

        <Route path='/' exact>

          <Redirect to='/quotes' />

        </Route>

        <Route path='/quotes' exact>

          <AllQuotes />

        </Route>

        <Route path='/quotes/:quoteId'>

          <QuoteDetail />

        </Route>

        <Route path='/new-quote'>

          <NewQuote />

        </Route>

        <Route path='\*'>

          <NotFound />

        </Route>

      </Switch>

    </Layout>

  );

}

export default App;

283. Implementing Pragrammatic Navigation (send to specific page)

Step 1 in NewQuote.js

import { useHistory } from 'react-router-dom';

import QuoteForm from '../components/quotes/QuoteForm';

const NewQuote = () => {

  const history = useHistory();

  const addQuoteHandler = (quoteData) => {

    console.log(quoteData);

    //send back to specific page

    history.push('/quotes');

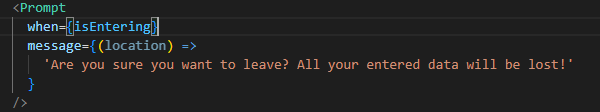
  };

  return <QuoteForm onAddQuote={addQuoteHandler} />;

};

export default NewQuote;

284. Preventing Possibly Unwanted Route Transitions with the “Prompt…”



Step 1 in QuoteForm.js

import { Fragment, useRef, useState } from 'react';

import { Prompt } from 'react-router-dom';

import Card from '../UI/Card';

import LoadingSpinner from '../UI/LoadingSpinner';

import classes from './QuoteForm.module.css';

const QuoteForm = (props) => {

  const [isEntering, setIsEntering] = useState(false);

  const authorInputRef = useRef();

  const textInputRef = useRef();

  function submitFormHandler(event) {

    event.preventDefault();

    const enteredAuthor = authorInputRef.current.value;

    const enteredText = textInputRef.current.value;

    // optional: Could validate here

    props.onAddQuote({ author: enteredAuthor, text: enteredText });

  }

  const finishEnteringHandler = () => {

    setIsEntering(false);

  };

  const formFocusedHandler = () => {

    setIsEntering(true);

  };

  return (

    <Fragment>

      <Prompt

        when={isEntering}

        message={(location) =>

          'Are you sure you want to leave? All your entered data will be lost!'

        }

      />

      <Card>

        <form

          onFocus={formFocusedHandler}

          className={classes.form}

          onSubmit={submitFormHandler}

        >

          {props.isLoading && (

            <div className={classes.loading}>

              <LoadingSpinner />

            </div>

          )}

          <div className={classes.control}>

            <label htmlFor='author'>Author</label>

            <input type='text' id='author' ref={authorInputRef} />

          </div>

          <div className={classes.control}>

            <label htmlFor='text'>Text</label>

            <textarea id='text' rows='5' ref={textInputRef}></textarea>

          </div>

          <div className={classes.actions}>

            <button onClick={finishEnteringHandler} className='btn'>Add Quote</button>

          </div>

        </form>

      </Card>

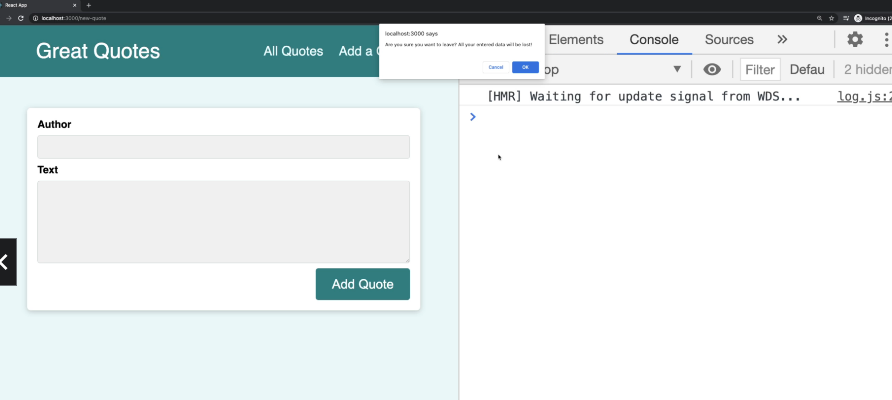
    </Fragment>

  );

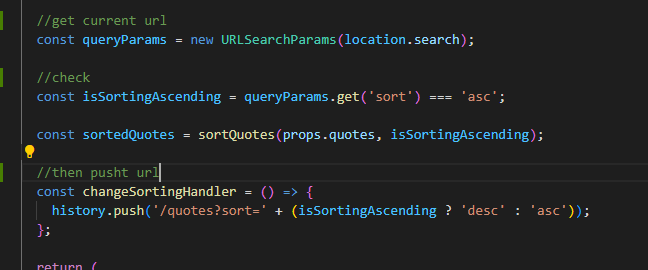
};

export default QuoteForm;

Step 2 check result



285. Working with Query Parameters



Step 1 in QuoteList.js

import { Fragment } from 'react';

import { useHistory, useLocation } from 'react-router-dom';

import QuoteItem from './QuoteItem';

import classes from './QuoteList.module.css';

const sortQuotes = (quotes, ascending) => {

  return quotes.sort((quoteA, quoteB) => {

    if (ascending) {

      return quoteA.id > quoteB.id ? 1 : -1;

    } else {

      return quoteA.id < quoteB.id ? 1 : -1;

    }

  });

};

const QuoteList = (props) => {

  const history = useHistory();

  const location = useLocation();

  const queryParams = new URLSearchParams(location.search);

  const isSortingAscending = queryParams.get('sort') === 'asc';

  const sortedQuotes = sortQuotes(props.quotes, isSortingAscending);

  const changeSortingHandler = () => {

    history.push('/quotes?sort=' + (isSortingAscending ? 'desc' : 'asc'));

  };

  return (

    <Fragment>

      <div className={classes.sorting}>

        <button onClick={changeSortingHandler}>

          Sort {isSortingAscending ? 'Descending' : 'Ascending'}

        </button>

      </div>

      <ul className={classes.list}>

        {sortedQuotes.map((quote) => (

          <QuoteItem

            key={quote.id}

            id={quote.id}

            author={quote.author}

            text={quote.text}

          />

        ))}

      </ul>

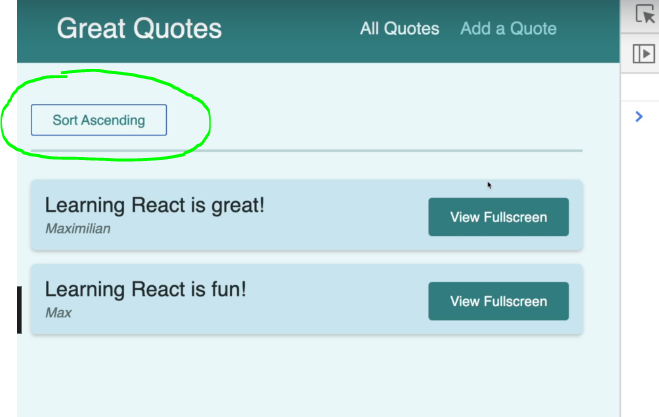
    </Fragment>

  );

};

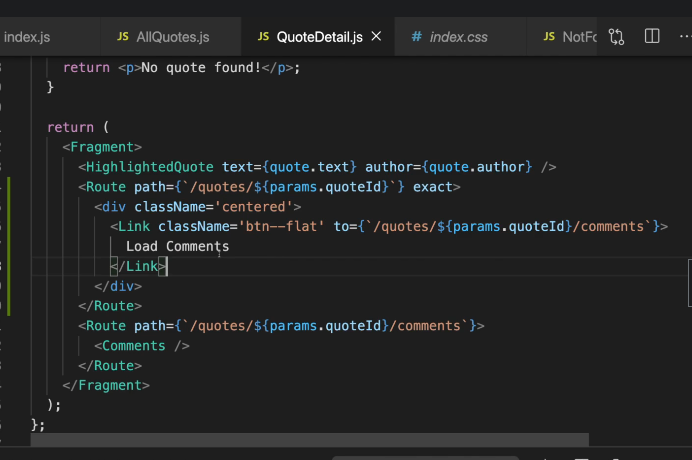
export default QuoteList;

Step 2 Check result

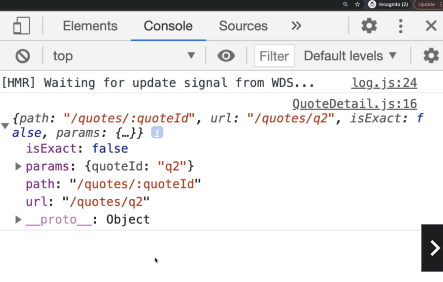


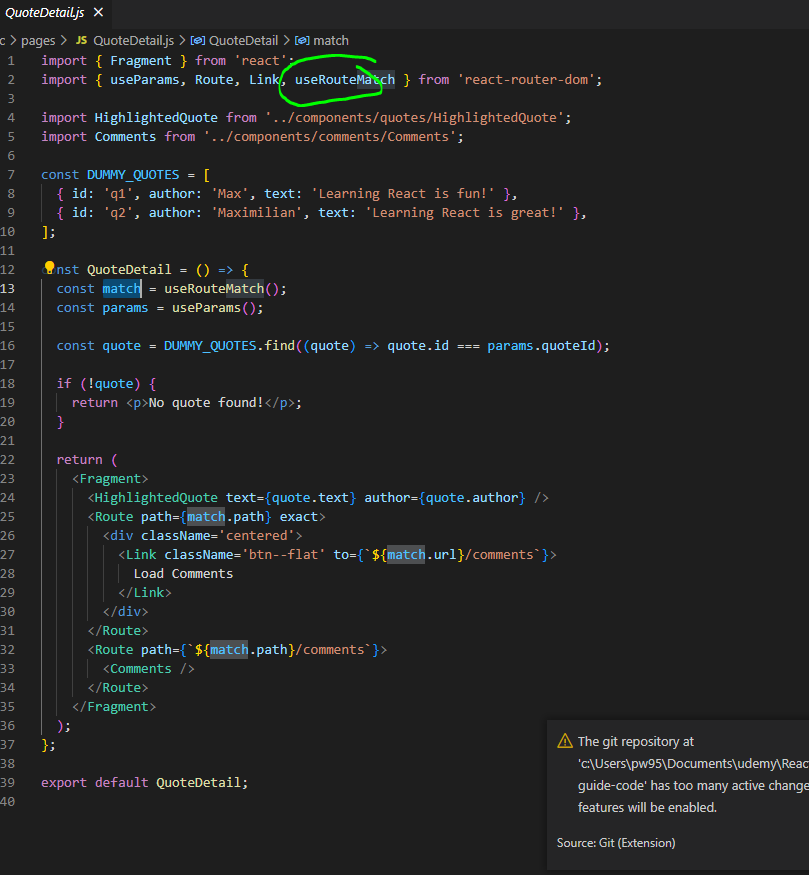
286. Getting Creative with Nested Routes

Step 1 in QuoteDetail.js



287. Writing More Flexible Routing Code (Reduce hard code url in children)





Step 1 in QuoteDetail.js

import { Fragment } from 'react';

import { useParams, Route, Link, useRouteMatch } from 'react-router-dom';

import HighlightedQuote from '../components/quotes/HighlightedQuote';

import Comments from '../components/comments/Comments';

const DUMMY\_QUOTES = [

  { id: 'q1', author: 'Max', text: 'Learning React is fun!' },

  { id: 'q2', author: 'Maximilian', text: 'Learning React is great!' },

];

const QuoteDetail = () => {

  const match = useRouteMatch();

  const params = useParams();

  const quote = DUMMY\_QUOTES.find((quote) => quote.id === params.quoteId);

  if (!quote) {

    return <p>No quote found!</p>;

  }

  return (

    <Fragment>

      <HighlightedQuote text={quote.text} author={quote.author} />

      <Route path={match.path} exact>

        <div className='centered'>

          <Link className='btn--flat' to={`${match.url}/comments`}>

            Load Comments

          </Link>

        </div>

      </Route>

      <Route path={`${match.path}/comments`}>

        <Comments />

      </Route>

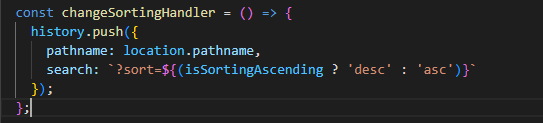
    </Fragment>

  );

};

export default QuoteDetail;

Step 2 in QuoteList.js (reduce complex)



import { Fragment } from 'react';

import { useHistory, useLocation } from 'react-router-dom';

import QuoteItem from './QuoteItem';

import classes from './QuoteList.module.css';

const sortQuotes = (quotes, ascending) => {

  return quotes.sort((quoteA, quoteB) => {

    if (ascending) {

      return quoteA.id > quoteB.id ? 1 : -1;

    } else {

      return quoteA.id < quoteB.id ? 1 : -1;

    }

  });

};

const QuoteList = (props) => {

  const history = useHistory();

  const location = useLocation();

  const queryParams = new URLSearchParams(location.search);

  const isSortingAscending = queryParams.get('sort') === 'asc';

  const sortedQuotes = sortQuotes(props.quotes, isSortingAscending);

  const changeSortingHandler = () => {

    history.push({

      pathname: location.pathname,

      search: `?sort=${(isSortingAscending ? 'desc' : 'asc')}`

    });

  };

  return (

    <Fragment>

      <div className={classes.sorting}>

        <button onClick={changeSortingHandler}>

          Sort {isSortingAscending ? 'Descending' : 'Ascending'}

        </button>

      </div>

      <ul className={classes.list}>

        {sortedQuotes.map((quote) => (

          <QuoteItem

            key={quote.id}

            id={quote.id}

            author={quote.author}

            text={quote.text}

          />

        ))}

      </ul>

    </Fragment>

  );

};

export default QuoteList;

288. Sending & Getting Quote Data via http (add use-http)

Step 1 clone source to get use-http.js and api.js

Use-http.js

import { useReducer, useCallback } from 'react';

function httpReducer(state, action) {

  if (action.type === 'SEND') {

    return {

      data: null,

      error: null,

      status: 'pending',

    };

  }

  if (action.type === 'SUCCESS') {

    return {

      data: action.responseData,

      error: null,

      status: 'completed',

    };

  }

  if (action.type === 'ERROR') {

    return {

      data: null,

      error: action.errorMessage,

      status: 'completed',

    };

  }

  return state;

}

function useHttp(requestFunction, startWithPending = false) {

  const [httpState, dispatch] = useReducer(httpReducer, {

    status: startWithPending ? 'pending' : null,

    data: null,

    error: null,

  });

  const sendRequest = useCallback(

    async function (requestData) {

      dispatch({ type: 'SEND' });

      try {

        const responseData = await requestFunction(requestData);

        dispatch({ type: 'SUCCESS', responseData });

      } catch (error) {

        dispatch({

          type: 'ERROR',

          errorMessage: error.message || 'Something went wrong!',

        });

      }

    },

    [requestFunction]

  );

  return {

    sendRequest,

    ...httpState,

  };

}

export default useHttp;

in api.js

const FIREBASE\_DOMAIN = 'https://react-prep-default-rtdb.firebaseio.com';

export async function getAllQuotes() {

  const response = await fetch(`${FIREBASE\_DOMAIN}/quotes.json`);

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not fetch quotes.');

  }

  const transformedQuotes = [];

  for (const key in data) {

    const quoteObj = {

      id: key,

      ...data[key],

    };

    transformedQuotes.push(quoteObj);

  }

  return transformedQuotes;

}

export async function getSingleQuote(quoteId) {

  const response = await fetch(`${FIREBASE\_DOMAIN}/quotes/${quoteId}.json`);

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not fetch quote.');

  }

  const loadedQuote = {

    id: quoteId,

    ...data,

  };

  return loadedQuote;

}

export async function addQuote(quoteData) {

  const response = await fetch(`${FIREBASE\_DOMAIN}/quotes.json`, {

    method: 'POST',

    body: JSON.stringify(quoteData),

    headers: {

      'Content-Type': 'application/json',

    },

  });

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not create quote.');

  }

  return null;

}

export async function addComment(commentData, quoteId) {

  const response = await fetch(`${FIREBASE\_DOMAIN}/comments/${quoteId}.json`, {

    method: 'POST',

    body: JSON.stringify(commentData),

    headers: {

      'Content-Type': 'application/json',

    },

  });

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not add comment.');

  }

  return { commentId: data.name };

}

export async function getAllComments(quoteId) {

  const response = await fetch(`${FIREBASE\_DOMAIN}/comments/${quoteId}.json`);

  const data = await response.json();

  if (!response.ok) {

    throw new Error(data.message || 'Could not get comments.');

  }

  const transformedComments = [];

  for (const key in data) {

    const commentObj = {

      id: key,

      ...data[key],

    };

    transformedComments.push(commentObj);

  }

  return transformedComments;

}

Step 2 in NewQuote.js

import { useEffect } from 'react';

import { useHistory } from 'react-router-dom';

import QuoteForm from '../components/quotes/QuoteForm';

import useHttp from '../hooks/use-http';

import { addQuote } from '../lib/api';

const NewQuote = () => {

  const { sendRequest, status } = useHttp(addQuote);

  const history = useHistory();

  useEffect(() => {

    if (status === 'completed') {

      history.push('/quotes');

    }

  }, [status, history]);

  const addQuoteHandler = (quoteData) => {

    sendRequest(quoteData);

  };

  return <QuoteForm isLoading={status === 'pending'} onAddQuote={addQuoteHandler} />;

};

export default NewQuote;

Step 3 in AllQuotes.js

import { useEffect } from 'react';

import QuoteList from '../components/quotes/QuoteList';

import LoadingSpinner from '../components/UI/LoadingSpinner';

import NoQuotesFound from '../components/quotes/NoQuotesFound';

import useHttp from '../hooks/use-http';

import { getAllQuotes } from '../lib/api';

const AllQuotes = () => {

  const { sendRequest, status, data: loadedQuotes, error } = useHttp(

    getAllQuotes,

    true

  );

  useEffect(() => {

    sendRequest();

  }, [sendRequest]);

  if (status === 'pending') {

    return (

      <div className='centered'>

        <LoadingSpinner />

      </div>

    );

  }

  if (error) {

    return <p className='centered focused'>{error}</p>;

  }

  if (status === 'completed' && (!loadedQuotes || loadedQuotes.length === 0)) {

    return <NoQuotesFound />;

  }

  return <QuoteList quotes={loadedQuotes} />;

};

export default AllQuotes;

Step 4: in NoQuotesFound.js

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

import classes from './NoQuotesFound.module.css';

const NoQuotesFound = () => {

  return (

    <div className={classes.noquotes}>

      <p>No quotes found!</p>

      <Link className='btn' to='/new-quote'>

        Add a Quote

      </Link>

    </div>

  );

};

export default NoQuotesFound;

Step 5: in QuoteDetail.js

import { Fragment, useEffect } from 'react';

import { useParams, Route, Link, useRouteMatch } from 'react-router-dom';

import HighlightedQuote from '../components/quotes/HighlightedQuote';

import Comments from '../components/comments/Comments';

import useHttp from '../hooks/use-http';

import { getSingleQuote } from '../lib/api';

import LoadingSpinner from '../components/UI/LoadingSpinner';

const QuoteDetail = () => {

  const match = useRouteMatch();

  const params = useParams();

  const { quoteId } = params;

  const { sendRequest, status, data: loadedQuote, error } = useHttp(

    getSingleQuote,

    true

  );

  useEffect(() => {

    sendRequest(quoteId);

  }, [sendRequest, quoteId]);

  if (status === 'pending') {

    return (

      <div className='centered'>

        <LoadingSpinner />

      </div>

    );

  }

  if (error) {

    return <p className='centered'>{error}</p>;

  }

  if (!loadedQuote.text) {

    return <p>No quote found!</p>;

  }

  return (

    <Fragment>

      <HighlightedQuote text={loadedQuote.text} author={loadedQuote.author} />

      <Route path={match.path} exact>

        <div className='centered'>

          <Link className='btn--flat' to={`${match.url}/comments`}>

            Load Comments

          </Link>

        </div>

      </Route>

      <Route path={`${match.path}/comments`}>

        <Comments />

      </Route>

    </Fragment>

  );

};

export default QuoteDetail;

Step 6: check result

289. Adding the “Comments’ Features

Step 1 add NewCommentForm.js

import { useRef, useEffect } from 'react';

import useHttp from '../../hooks/use-http';

import { addComment } from '../../lib/api';

import LoadingSpinner from '../UI/LoadingSpinner';

import classes from './NewCommentForm.module.css';

const NewCommentForm = (props) => {

  const commentTextRef = useRef();

  const { sendRequest, status, error } = useHttp(addComment);

  const { onAddedComment } = props;

  useEffect(() => {

    if (status === 'completed' && !error) {

      onAddedComment();

    }

  }, [status, error, onAddedComment]);

  const submitFormHandler = (event) => {

    event.preventDefault();

    const enteredText = commentTextRef.current.value;

    // optional: Could validate here

    sendRequest({ commentData: { text: enteredText }, quoteId: props.quoteId });

  };

  return (

    <form className={classes.form} onSubmit={submitFormHandler}>

      {status === 'pending' && (

        <div className='centered'>

          <LoadingSpinner />

        </div>

      )}

      <div className={classes.control} onSubmit={submitFormHandler}>

        <label htmlFor='comment'>Your Comment</label>

        <textarea id='comment' rows='5' ref={commentTextRef}></textarea>

      </div>

      <div className={classes.actions}>

        <button className='btn'>Add Comment</button>

      </div>

    </form>

  );

};

export default NewCommentForm;

Step 2 in Comment.js

import { useState, useEffect, useCallback } from 'react';

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

import classes from './Comments.module.css';

import NewCommentForm from './NewCommentForm';

import useHttp from '../../hooks/use-http';

import { getAllComments } from '../../lib/api';

import LoadingSpinner from '../UI/LoadingSpinner';

import CommentsList from './CommentsList';

const Comments = () => {

  const [isAddingComment, setIsAddingComment] = useState(false);

  const params = useParams();

  const { quoteId } = params;

  const { sendRequest, status, data: loadedComments } = useHttp(getAllComments);

  useEffect(() => {

    sendRequest(quoteId);

  }, [quoteId, sendRequest]);

  const startAddCommentHandler = () => {

    setIsAddingComment(true);

  };

  const addedCommentHandler = useCallback(() => {

    sendRequest(quoteId);

  }, [sendRequest, quoteId]);

  let comments;

  if (status === 'pending') {

    comments = (

      <div className='centered'>

        <LoadingSpinner />

      </div>

    );

  }

  if (status === 'completed' && loadedComments && loadedComments.length > 0) {

    comments = <CommentsList comments={loadedComments} />;

  }

  if (

    status === 'completed' &&

    (!loadedComments || loadedComments.length === 0)

  ) {

    comments = <p className='centered'>No comments were added yet!</p>;

  }

  return (

    <section className={classes.comments}>

      <h2>User Comments</h2>

      {!isAddingComment && (

        <button className='btn' onClick={startAddCommentHandler}>

          Add a Comment

        </button>

      )}

      {isAddingComment && (

        <NewCommentForm

          quoteId={quoteId}

          onAddedComment={addedCommentHandler}

        />

      )}

      {comments}

    </section>

  );

};

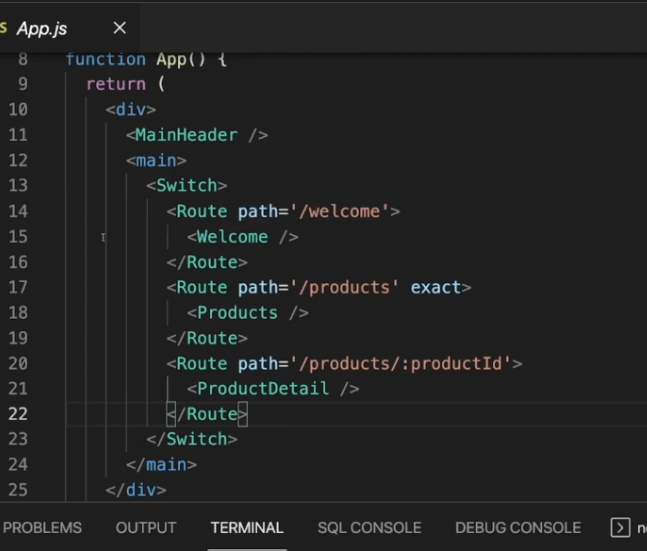
export default Comments;

290. Upgrading To React Router V6 (https://reactrouter.com/en/main)

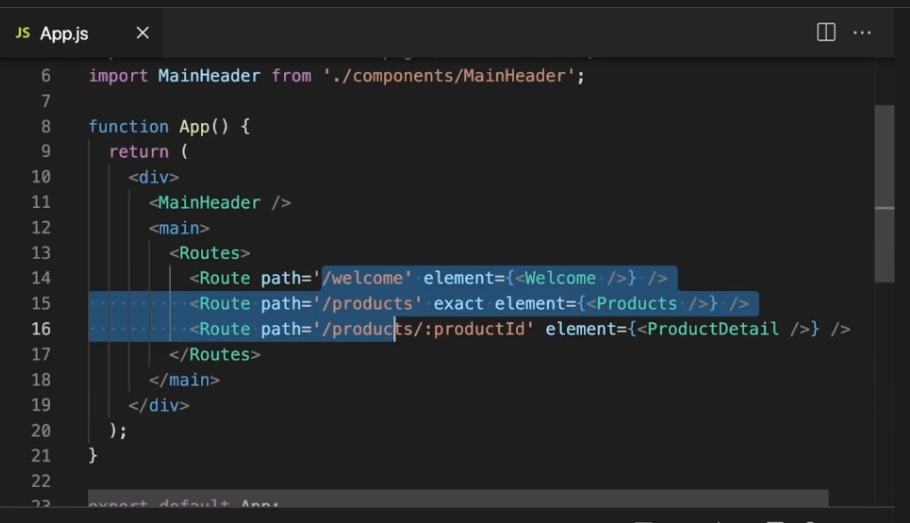
Step 1 Run npm install react-router-dom@6 when use

Step 2 in app.js (use Routes instead use Switch)

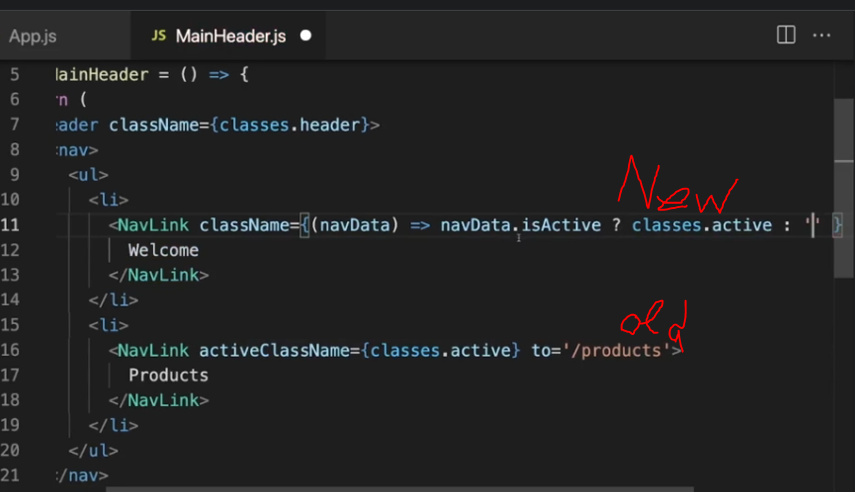
Old



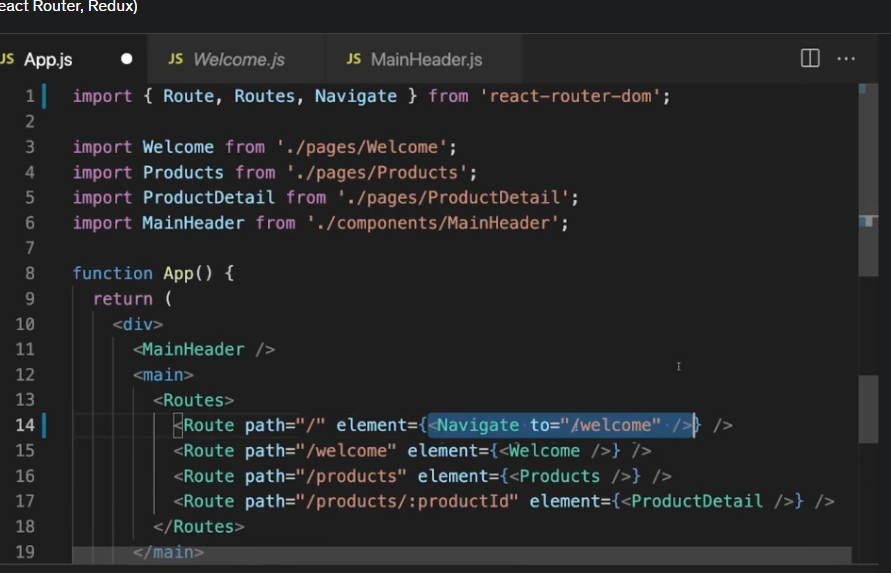
new



Step 3 some change in NavLink

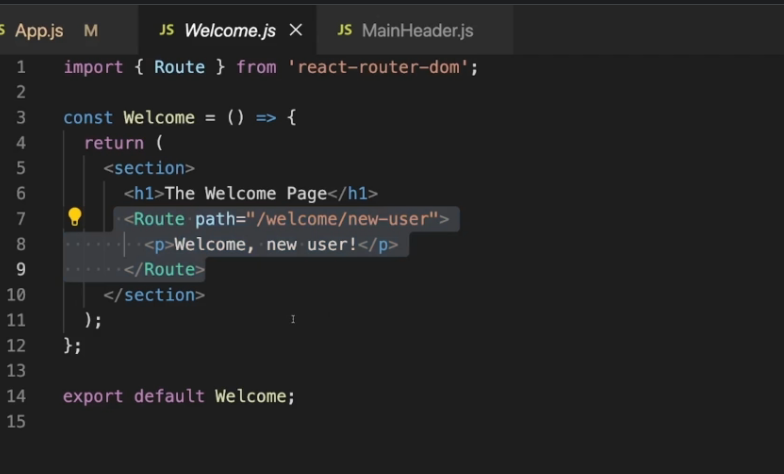


Step 4: change Redirect to Navigate

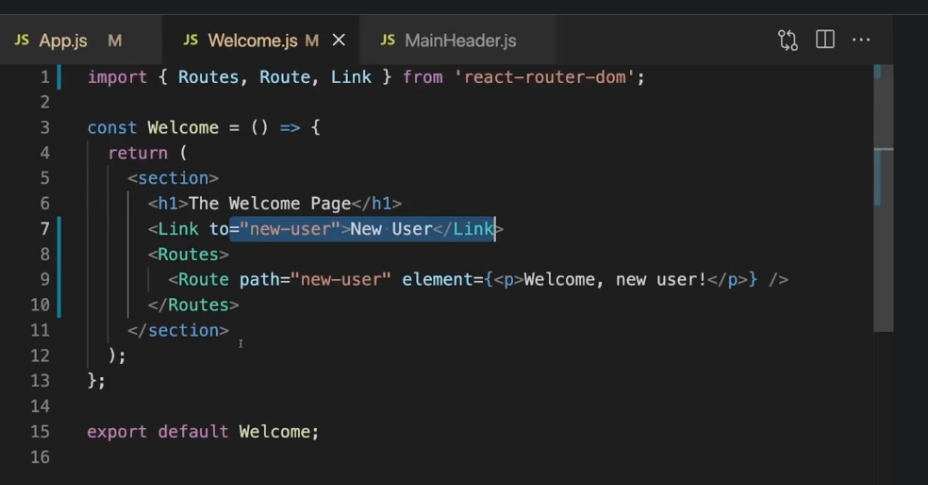


Step 5: change nested url ()

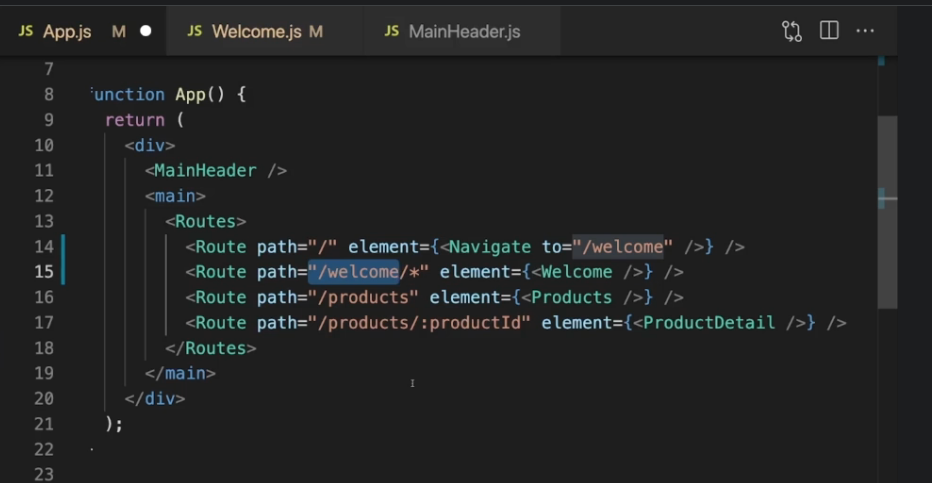
Old



New

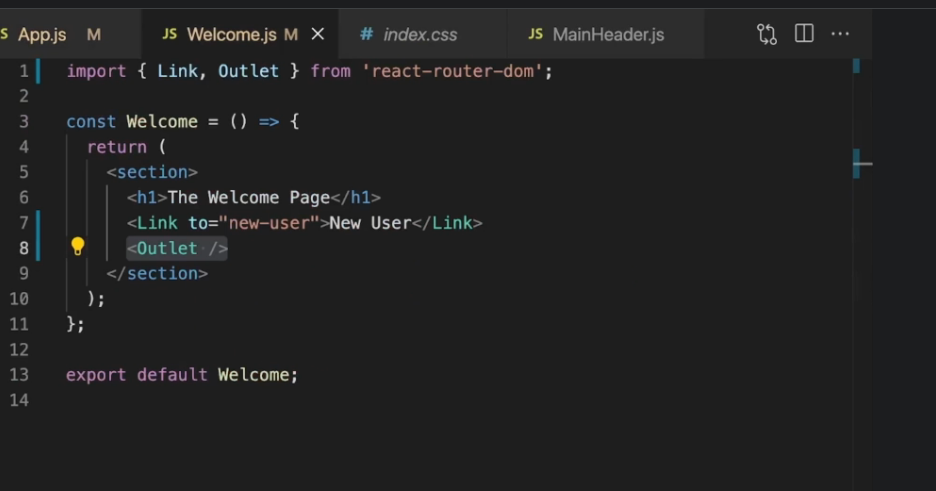


Then in app.js

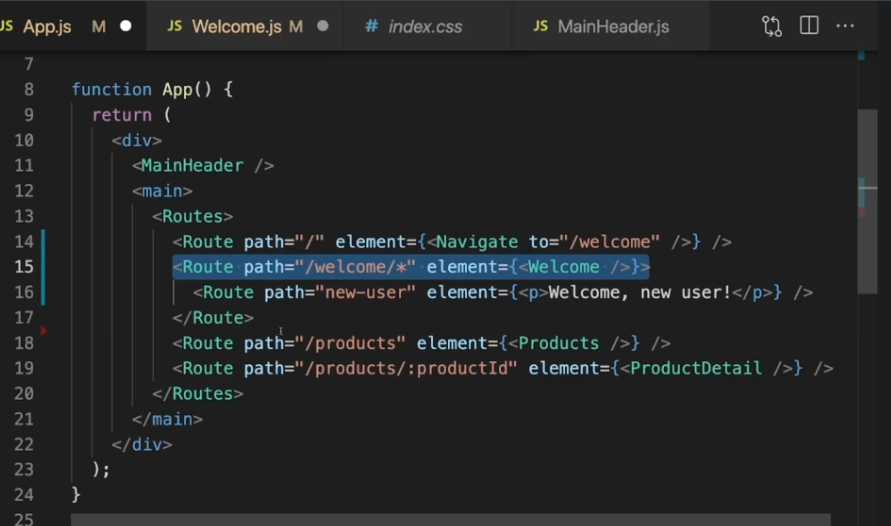


OR

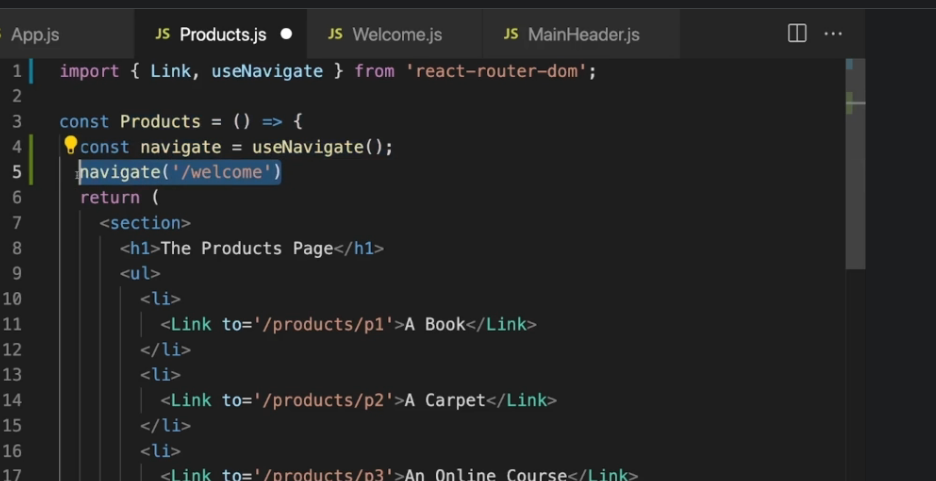
In Welcome.js



In App.js



Step 6: use Navigate instead use UseHistory



Step 7 Prompt (no more feature to Prompt)

291. React Route 6.4 (no source, just image) (Advanced)