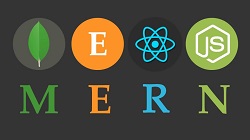
*MERN Full Stack Development*

**

**Class 8:**

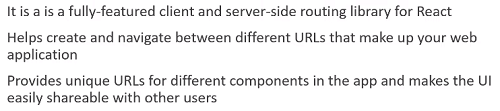
**React Router**

**React Fundamental and Advanced Topics**

1. React Hooks
2. React Redux
3. React Formik
4. React Render
5. React TypeScript
6. React Storybook
7. React Query
8. React Styled Components
9. React Table

**What is React Router?**

* It is a fully featured client and server side routing library for react.
* If we building a medium or large scale react app react router is must have package, we will see the following things.
  + Configuring routes
  + Navigation on button click
  + Navigate programmatically
  + Dynamic Routes
  + Nested Routes
  + Route Parameters
  + Lazy Loading
  + React Authentication



**Pre-requisite**

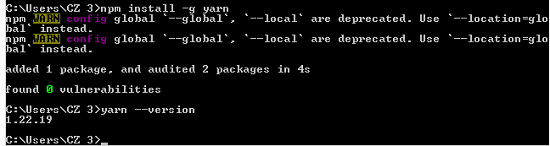
* React Fundamentals
* React Hooks

**Router installation and setup**

* First we are going to setup our project and start using react router.
* So first create a new react app or use the old react app by using npx create-react-app command.
* Then 2nd thing install the react router package so in the terminal write the command “yarn add react-router-dom@6 or the command npm install react-router-dom@6 hit enter it will start the installation.

yarn: The term 'yarn' is not recognized as the name of a cmdlet, function, script file, or operable program.

* If you see this type of error above you have to first install the yarn globally.and after install we can yarn –version.

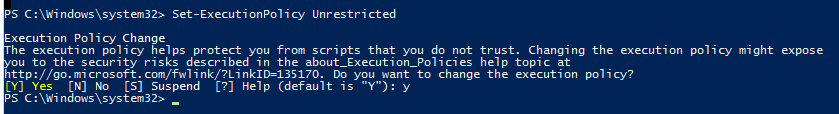
****

* But if you still see error like below. Follow below steps

# yarn cannot be loaded because running scripts is disabled on this system

* Go to start type power shell and run powershell as administrator and type the following command.

**Run powershell as administrator and run this command: `Set-ExecutionPolicy Unrestricted`**

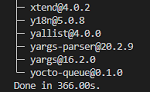


* Now again try to install react router by using this command yarn add react-router-dom@6

**React Classes\myproject> yarn add react-router-dom**

**yarn add v1.22.19**

**info No lockfile found.**

****

* This should install react router version 6 in our project just open up the package.json file and check the dependencies.

  "react": "^18.2.0",

    "react-dom": "^18.2.0",

    "react-router-dom": "^6.3.0",

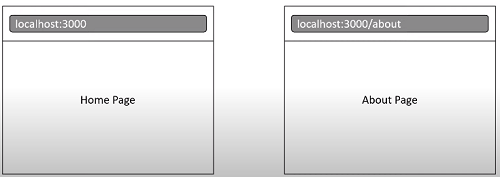
    "react-scripts": "5.0.1",

    "web-vitals": "^2.1.4"

* So we have successfully install react router and now next configure routes and render different components or different urls in the browser.

**Configuring Routes**

* Now first we are going to learn how to configured routes with react router, we are going to setup three routes.
* The first route is the home route, if the user navigates to localhost: port 3000 they should be able to see the home page and if they visit the localhost:3000/about they should be able to see the about page.
* In our src folder we have index.js file where we render the app component to the Dom, the app component is present in app.js
* And it contains a very simple UI, so the first step to configuring routes with react router is to connect the url in the browser with our react app.
* For that react router provides a component called browser router with which we need to wrap our entire app so in index.js file.

****

* At the top the imports area “import { BrowserRouter} from ‘react-router-dom’, this is the react router component named “browser router”.

Index.js

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

import reportWebVitals from './reportWebVitals';

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

* Next wrap the app component with browser router

  <React.StrictMode>

    <BrowserRouter>

      <App />

    </BrowserRouter>

  </React.StrictMode>

* What this allows us to do is use all the features from react router within the app component tree, now go to app.js file and configure the routes.
* First we remove all the imports and empty the app component function.

function App() {

  return (

    <div className="App">

      <header className="App-header">

* But before we configure the routes we need the components that need to be rendered for the app routes, / about and /contactus in the url.
* Now in the components folder we are going to create 3 new files home.js , aboutus.js and contact.js
* In the home.js file create a functional arrow function which renders h1 heading “Home page” do the same thing in aboutus and contactus.

const Home = () => {

  return (

    <div>

        <h1>Home page</h1>

    </div>

  )

}

* So now we have the 3 views so let’s configure the routes.
* For the route configuration we need 2 components from react router.
* So in app.js file just import the { Routes, Route } from ‘react-router-dom’.

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

function App() {

* And in the Component JSX just add routes component and within routes component we defined the individual route using the route component.
* And on the route component we specify the 2 props the first prop is path which reflects the path in the url, our first route is the root of the app.
* And which is localhost:3000 port & that is denoted by forward slash “/”.
* So value to path prop is a forward slash, we then tell react router what is the element that needs to be rendered when the url matches this path.
* In our case it would be the home component so we add the 2nd prop which is element and to this we assign the home component.
* And make sure to import the home component at the top save and check in the browser we should see the text home page being displayed.

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

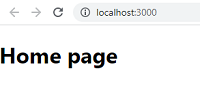
import Home from './mycomponents/Home';

<Routes>

            <Route path='/' element={<Home />}></Route>

        </Routes>

* So this is the JSX home component so our first route is working.



* Let’s configure the 2nd route now invoke the route component once again.we are going to set path to “about” and element going to be equal to “about” component & make sure is import it about component at top.

  <Routes>

            <Route path='/' element={<Home />}></Route>

            <Route path='about' element={<Aboutus />}></Route>

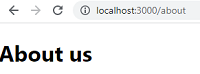
        </Routes>

* So we are asking react router to render about component when the path in url is about, of course in both these cases route can be self-closing tag.

            <Route path='/' element={<Home />} />

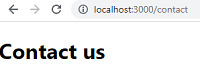
            <Route path='about' element={<Aboutus />} />

* If we head back to the browser and type locahost:300/about and we can see the aboutus page being rendered.



* So our route configuration is working well, this is pretty much how we configure routes with react router.
* Now do the same thing for contact us page component.

  <Route path='contact' element={<Contactus />} />



* So first step we wrap the root component of the app with browser router from react router package done in index.js file.
* Step 2 create components views that need to be rendered at different urls path we created the home, aboutus & contactus components.
* Step 3 we configure the routes using the routes and route components from react router done in app.js component file.
* Routes at the top level which can contain multiple route components each route will accepts a path prop which corresponds to the path in the browser url and the corresponding react element to render when the path is matched we have set home component for the root of the app and about us and contact us component for the about and contact path.
* But what we might have noticed is that we navigate to the different pages by entering the url in the browser address bar and this off course the regular user would navigate in a web application.
* We have a UI element like a link which the user can click to navigate to a different route or the user could also be navigated programmatically to a different route after an action has completed for example.
* Let’s see how to do this same with the react router.
* Next we will see how to navigate to different routes using an element in the UI, now make a links/navigation.

**Routes Links**

* Let’s see how to navigate to different routes using an element in UI.
* Let’s use a navbar in app component with 3 links “Home page”, “About us page” & “Contact us”, when click on these links we should be able to navigate between two routes, in component create new file.
* Nav.js, inside file define navbar using rafce and navbar is component name, put nav tag, inside we need 3 clickable elements to navigate between routes, to navigate to other roués react router provides us with the link component, so at the top import component react router dom.

import React from 'react'

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

const nav = () => {

  return (div></div>

  )

}

export default nav

* Inside nav tag invoke link component that renders an anchor element in DOM so we define text similar to anchor text Home, Aboutus, Contactus.
* Instead of “href” attribute we specify 2 props “to” this prop we assign path of configured route for home is going to be forward “/”.
* And for about is to be “/aboutus” see below code & include this app.js

 <nav>

        <Link to='/'>Home</Link>

        <Link to='/aboutus'>Aboutus</Link>

<Link to='/contactus>Contactus</Link>

    </nav>

* In JSX wrap with fragments <>, </> & include nav component make & import nav component at top, save see code & check in browser.

import Nav from './mycomponents/Nav';

<>

       <Nav />

        <Routes>

            <Route path='/' element={<Home />} />

            <Route path='aboutus' element={<Aboutus />} />

            <Route path='contactus' element={<Contactus />} />

        </Routes>

      </>

* We should be able to see home,aboutus, contact us links, use styling to this navbar to make it looks good, go to index.css file.
* Add some styling to nav tag & nav anchor tag see code & go to browser.

nav {

    background-color: darkgoldenrod;

    padding: 18px 36px;}

nav a {

  margin-right: 15px;

  color:white;

  font-family: monospace;

  font-size: 26px;

  font-weight: bold;

  text-decoration: none;}

nav a:hover {

  text-decoration: underline;

   color:orange;

}

****

* We are currently at localhost:3000 & hence we see a homepage, if we click on about link we are navigated to /aboutus & about us page is rendered.
* If we inspect link tag we can see it is an anchor tag with href attribute, so this is how we make use of link component for client side navigation.
* It is navigating within application if we want to navigate to an external website we can use plain old html anchor tag.
* In route config in app.js we have specified just aboutus as path whereas in nav component on link component we have specified /aboutus.
* If we remove this forward slash “/” and see in browser it still works same.
* However behavior can change depending on route we are currently rendering link component from & that is what is called as relative path.
* For now we are going to work with absolute routes but path will begin with a forward slash “/about” & match same route in route config.

**Active Links**

* We create 3 links to navigate back & forth between home, aboutus, contact us page however all 3 links appear same irrespective of which route we are currently viewing, in practical web applications.
* It is common to style link corresponding to active route in a different way.
* Let’s style active link, react router provides another component called “nav link” which knows whether or not it is active link.
* We can use that component to style active link to our nav, so within nav.js at top instead of importing link import nav link & replace with same in JSX.

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

const Nav = () => {

  return (

    <nav><NavLink to='/'>Home</NavLink>

        <NavLink to='/about'>Aboutus</NavLink>

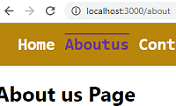
* What is special about this navlink is component though is that by default it receives an active class when link is current route goes to browser & see.

C:\Users\CZ 3\Desktop\Untitled.png

* Select current link like “home” & inspect element, we can see class = “active” on home link & if click on about & class is applied to about link.
* Now add class in our css, so go to index.css & apply following css.

nav a.active {

text-decoration: overline; color: rebeccapurple; font-weight: bolder;



* So we apply styling to active class on anchor within nav tag.
* But we don’t want to use plain CSS to style our react app, we now using a CSS in JS solution, & we also see how to make use of active link with style prop. So we just comment out active nav css in index.css.
* And in Nav.js we add style prop on all 3 navlinks & set it to a function “myNavLinkStyle”, define this function by using arrow function.
* NavLink component will set a property called “isActive” which can be de-structured from this function parameter, so curly braces to de structure a property called “isActive”, isActive will be set to true if link is the current route, we can use this to conditionally apply styles.
* Apply the style with JavaScript, so we return an object where we set font weight depending on isActive, it is an active link font weight will be “bold” else “normal” used ternary operators, Also text decoration depending on active flag is “overline” if is indeed active route if not is “underline”.

const myNavLinkStyle = ({ isActive }) => {

      return {

          fontWeigth: isActive ? 'bolder' : '500',

          textDecoration: isActive ? 'overline' : 'underline',

          color: isActive ? 'orange' : 'white',

      }}

   <NavLink style={myNavLinkStyle} to='/'>Home</NavLink>

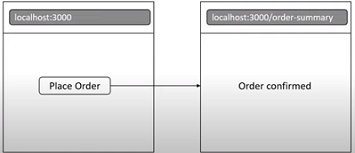
* Save file and go to browser result will be same, so to set a different style on active makes use of navlink component we can either use of active class that is applied or “isActive” Boolean flag injected into style function.

C:\Users\CZ 3\Desktop\Untitled.pngC:\Users\CZ 3\Desktop\Untitled.png

* We used navlink component instead of link component b/c navlink component is specifically meant for building components like a navbar or breadcrumbs or set of tabs where we would like to highlight current selected item & provide useful context with screen readers.
* If we want to navigate to routes from other parts of app so we can use link component to use as we wouldn’t want active class being applied when it is not needed, so we now understand how to navigate to different routes on click of UI element, next we will see how to navigate programmatically.

**Navigation with Programming Logics**

* We have seen how to navigate to different routes using link & navlink components, for better performance we might need to programmatically navigate to a particular route e.g. We placing an order on amazon/ebay.
* If the form submission is successful.
* we would be navigated to order confirmation page the scenario can hold good for any type of form submission as well & is needed in one or other application we are going to build let’s see this in react router.



* We are amusing a scenario & implement a button click handler where we programmatically navigate to a different route in our application.
* For our e,g, on click of a place order button in homepage let’s navigate to order summary page, so we are going to create a functional component.
* Named it “OrderSummary” that reads text “orders have been confirmed”.

const OrderSummary = () => {

  return (

    <div>Order Have been received!!</div>

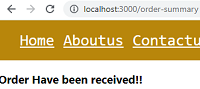
  )

* In app.js add this new route path is going to be order-hyphen summary.
* And element is going to be equal to “OrderSummary” component.

     <Route path='order-summary' element={<OrderSummary />} />

         </Routes>

* Go to browser and navigate to localhost:300/order-summary, we should see text “order have been confirmed”, we want this page navigate to on click of a button or link it will happens after an event is successful



* Add button in home page that says place order, in home.js in JSX add fragments & add a button, we want to do is on click of this button to navigate to order summary page, to navigate programmatically react router provides the “useNavigate” hook so import this at the top.

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

* We can now invoke this hook within component “useNavigate” this hook returns a function which we going to call as navigate. See below
* Using this navigate function we can now navigate to order summary route on click of button, so pass navigate function equal to arrow function & passing in path for route which is navigate(‘order-summary’).
* Save & test in browser, on home page click on place order button.

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

const Home = () => {

  const orderNavigate = useNavigate()

  return (

    <><h1>Home page</h1>

 <button onClick={() => orderNavigate('order-summary')}>Order Place</button></>

  )

* We navigated to order summary page this is navigated programmatically.

C:\Users\CZ 3\Desktop\Untitled.png

* Now Implementing back button programmatically is also done with use Navigate hook, in order-summary component add back button.
* Wrapped in fragments or div & add button “Back to homepage” & repeat same steps as before import useNavigate call hook & assign it to a const “backNavigate” which is function & when click on button call backNavigate & we want to go back to homepage pass value of “-1”.

const OrderSummary = () => {

  const backNavigate = useNavigate()

  return (

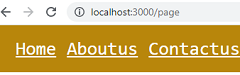
    <div><h3>Order Have been placed!!</h3>

        <button onClick={() => backNavigate(-1)}>Back to Homepage</button>

* Head back to browser, when click on “back” button we navigated to home.
* React router provides use navigate hook to navigate programmatically, call hook in component & use return function to navigate passing in either a path or -1 number that indicated a home page.

**No Matching Route**

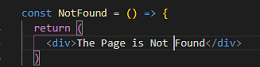
* Learn a no match route in react application, if we trying to a route that is not configured e.g. localhost:3000/page, we don’t see UI below navbar.



* If we see a console we can have a warning “no routes matched location”.

C:\Users\CZ 3\Desktop\Untitled.png

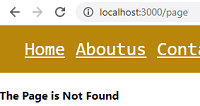
* This scenario is not favorable from a user point of view, a user might assume that app is still loading or there is an error in /page route.
* What would be better if we inform user that URL does not match any route in our application, let see how to do that?
* Create new component that should be rendered for URLs that don’t match any of that configured routes, named it NotFound.js.
* In that functional component file renders text “The Page is not found!!”.



* Let see how to handle no match route go to app.js add new route its path = asterisk or star & this has a special meaning in react router.
* This route will match only when no other routes found, for element prop we are specifying “NotFound” component save & check in the browser.

     <Route path='\*' element={<NotFound />} />

* Now try to navigate /page , we can see page not found text corresponding to the no match component this much better user experience.

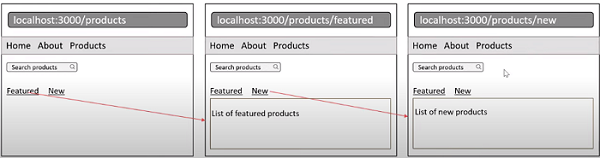


**Nested Routes**

* Nested routes feature with react router; we have seen react router helps us navigate across pages in our application.
* We should know that react router also helps to switch b/w a portion of view inside page, e.g. if our application consider a 3rd navigation link “products”, when we click on that link we are navigated to “/products”.



* This renders a new products page now this is something we are already familiar with but now within this products page we have something special, we have a search bar for search a product & within search bar we have 2 more links “featured” & “new”, if we click on featured url udpates to /products/featured/ the search & links will state in place.
* And only UI below link will change & render a list of featured products.
* If we click on “new” the URL changes to localhost:3000/products/new.



* The search bar & links will stay in place but list of featured products is switched with UI that renders list of new products.
* The only a portion of UI changes based on route to achieve this we make use of nested routes, let start step 1 configure a new route product page that add link in navbar, create file products.js, this renders a search input.

const Products = () => {

  return (

    <div> <input type="search" placeholder='Enter search product...' />div>

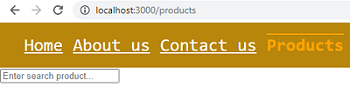
* Add this component in app.js so route path = ‘products’ & element = { component “<Products> we have just defined see below.

<Route path='products' element={<Products />} />

* In nav.js add a 4th link /products copy navLink about change to products.

<NavLink style={myNavLinkStyle}  to='/products'>Products</NavLink>

* In browser we should have products link when click it open product page.



* Step 2 add another set of navigation links with product component wrapped it fragments & add new tag for links we need link component from react router import {Link} from react router.

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

const Products = () => {

* The 1st link is featured products & 2nd link is latest products on link component we specify to prop 1st is to=’featured’ 2nd is to=’latest’.
* Make sure don’t include the “/” forward slash for nested route, this is Relative Paths (without ‘/’) which we will learn about later on a class.

<>

        <div>

            <input type="search" placeholder='Enter search product...' />

        </div>

        <nav>

            <Link to='featured'>Featured Products</Link>

            <Link to='latest'>Latest Products</Link>

        </nav>

    </>

* Save file & see in browser, we should see both links but same design of primary navbar so let’s make the style more specific.



* In index.css change nav styling to a class selector “.primary-nav” & also assign this class in nav.js & in nav tag add class = “primary-nav”.
* Index.css

.primary-nav {

    background-color: darkgoldenrod;

nav.js

    <nav className='primary-nav'>

        <NavLink style={myNavLinkStyle} to='/'>Home</NavLink>

* Step 3 creates 2 new components that need to be rendered for featured products & latest products & also configure new routes.
* In these components that read corresponding text see code.

const FeaturedProducts = () => {

  return (

    <div><h3>List of Featured Products</h3></div>

const LatestProducts = () => {

  return (

    <div> <h3>List of Latest Products</h3></div>

* Now configure new routes for these 2 components more specifically configure nested routes, both components have to be nested within products route, so for nesting we change route from self-closing tag to 1 that has a closing tag, so opening & closing route tags.
* And inside these tags we define 2 routes for route path=’featured’ & element = ‘{<FeaturedProducts />} similarly for latest products see code.

  <Route path='products' element={<Products />}>

                <Route path='featured' element={<FeaturedProducts />} />

                <Route path=latest element={<LatestProducts />} />

            </Route>

* What is special about nested routes is that react router automatically forms full path to children routes so featured is actually /products/ featured & latest is /products/latest, we have configured routes to render child component within parent component that is featured products or new products within products component.
* However products page still doesn’t know where to render child components for that react router provides an outlet component.
* So import it {Link, Outlet} & invoke it below nav tag & that is it sees below.

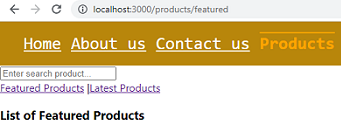
import { Link, Outlet } from 'react-router-dom'

const Products = () => {

        </nav>

        <Outlet />

* Let’s test it out in browser click on products & the URL changes to /products products component is also rendered now click on featured the URL changes to /products/featured & list of featured products showed.



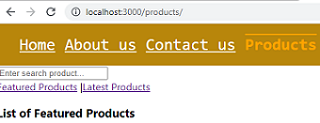
* Click on “latest” & URL changes to /products/latest & latest products component is rendered, we are able to change only a portion of UI in same page whenever URL changes this is how nested routes feature works with react router, so we 1st created products component & configured a route.
* We then added a link in nav to navigate to products page we then created 2 more components for featured & latest products.
* This time we configured routes to these components as nested routes we did that by including route component within route component.
* The parent route component is products’ the child component path will automatically have parent path as a prefix so new routes becomes products/feature & 2nd products/latest.
* Parent component needs to know what to do with these child components in routes tree for that we use outlet component from react router.
* The outlet component renders component corresponding to matching child route from parent list of routes, nested components as we can see is very powerful, 1 of common use cases of doing this is have a common layout for a featured in our application, the parent route will render layout component if we can call it that & within layout component use outlet component to render different child components.

**Index Route**

* So we have seen product page which consists of 2 nested routes “featured products” & “latest products”, child routes render only when URL /products/featured or /products/latest, sometimes we might want of child routes to render at parent route level, If we navigate just products.
* And URL read /products, we might still want to print list of featured products that can be done using index route in router.
* The index route is also a nested route, so inside products route add a new route component, we don’t need to specify path prop instead we specify a prop called index, this index route will share path of parent route which is products & specify element prop just like other routes.
* Since we want “featured products” to render the element will be featured products component, save & see in browser & navigate to products page.

<Route path='products' element={<Products />}>

                <Route index element={<FeaturedProducts />} />



* The URL is /products but we still see featured products child component.
* If we click on featured link it remains same & we can also navigate to latest products, so our index route is working as expected.
* So we have nested routes & we want a route to be rendered at parent URL make use of an index route, index route contain index prop instead of path prop next we will be going to see dynamic routes with react router.

**Dynamic Routes**

* Learn dynamic route, assume we are building an admin panel & we need user listing & details page, if user navigates to /users we should display a list of 5 users, if user navigates to /users/id followed by id of that user.
* We need to display details about that individual user, e.g. if user navigate /users/1 we should display details about 1st user.
* Similarly if user navigates to /users/2 we need to display details 2nd user.



* New functional component file “Users,js”, this component will display list of users, use heading for all users, configure a route for this component.

const Users = () => {

  return (

    <div>h2>User List</h2>

           <h3>User 1</h3>

           <h3>User 2</h3>

           <h3>User 3</h3>

* In app.js the path prop is equal to users & elements = users component.

    <Route path='users' element={<Users />} />

* Type localhost:3000/users in URL we should see list of users in this page.

C:\Users\CZ 3\Desktop\Untitled.png

C:\Users\CZ 3\Desktop\Untitled.png

* Now details page, we need a detailed view for users so create UserDetails.
* A functional component user details has to be rendered for all 5 different urls, like /users/1, /users/2 and so on, let add this route in app.js.

const UserDetails = () => {

  return (

    <div>Details About User</div>

* Route path = users/1 element = UserDetails component, similarly we are going to have 4 more routes for /users/2 etc see code below.

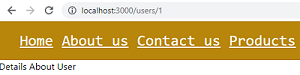
            <Route path='users/1' element={<UserDetails />} />

            <Route path='users/2' element={<UserDetails />} />

            <Route path='users/3' element={<UserDetails />} />

            <Route path='users/4' element={<UserDetails />} />

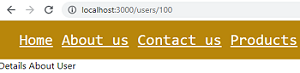
* Now go to browser and see /users/1 we see users details page & same is the case with /users/2 or others, it works but this is not feasible solution.



* If we have 100 or 1000 of users we would need to configure a 100 or 1000 different routes, the correct solution is to use dynamic route segments.
* In our scenario user id which can be 1,2,3 & so on should be a dynamic value & for such a value we specify what is termed as a URL parameter in react router, so instead of users/1 we specify users/:userId.
* Remove others all routes, this user id param will match any value as long as pattern is the same that is URL in browser is /users/ any value.

       <Route path='users/:userId' element={<UserDetails />} />

* Go to browser & navigate to /users/1 we see user details page, it will works for users/2, or user/100 or any number.



* So when we have to work with list & details routes.
* Dynamic routes are what we need; note the 1 point user id can be any string & not a number, so in browser I could type /users/admin.

C:\Users\CZ 3\Desktop\Untitled.png

* And it would render same page , but this dynamic route we have configured we would also need a fixed path, so create new component admins.js , use functional component which show admin text see below.

const Admins = () => {

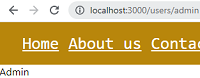
  return (

    <div>Admin</div>

* In routes add another route path = users/admin & element is Admins component, if we navigate to users/admin will user details page be rendered or admins page rendered go to browser and find out.

    <Route path='users/admin' element={<Admins />} />

* So localhost:300/users/admin & we can find admin component is here.



* Even though we have a dynamic route where user id can be anything react router is smart enough to 1st match route that is more specific.
* So if we navigate to users/admin react router 1st try to find a matching route only if that is not found will it match dynamic route.
* Dynamic route can be nested as well; since 2 routes we have just configured have users as prefix we can nest them.
* The child component would be rendered within parent component.
* So user’s parent route will now have opening closing tags & the 2 routes are nested inside from path prop thought we need to remove users.

<Route path='users' element={<Users />}>

              <Route path=':userId' element={<UserDetails />} />

              <Route path='admin' element={<Admins />} />

            </Route>

* And in users component we need to add outlet component for rendering child, so import outlet & invoke it below the list of users see code.

import { Outlet } from 'react-router-dom'

const Users = () => {

       <h3>User 4</h3>

           <Outlet /></div>

* In browser & our nested dynamic route is working /users/admin so admin component is rendered, /users/1 user details component is rendered.

C:\Users\CZ 3\Desktop\Untitled.png C:\Users\CZ 3\Desktop\Untitled.png

C:\Users\CZ 3\Desktop\Untitled.png C:\Users\CZ 3\Desktop\Untitled.png

* Take away points, 1st when we dealing with a list details pattern or if route parameter can vary in value we use of dynamic routes.
* Specify URL param denoted by “:” colon prefix in path 2nsd point react router will always try to match route that is more specific before trying to match a dynamic route so /admin before /users id.
* 3rd is possible to have nested dynamic routes.
* Routing Class will be continuing in Class 9.