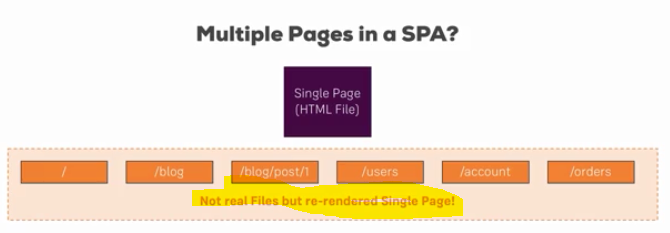
# Routing

Routing : Multi-page feeling in Single-page application.

Routing module is not built in the core of react module. React at its core is just a component creation library.

react-router is a powerful routing library built on top of React.

Routing is about being able to show different pages to the user. **we don't really have different files but simply we re-render parts off that single page or maybe the entire single page depending on which path the user navigated to in our application.**



**This is what routing is about, parsing this path, so the path after our domain and showing the appropriate jsx or component code in our app.**

for that we're going to use a router package to add such a functionality so that we don't have to parse that path on our own.

This is the idea behind routing, load different code, conditional jsx or component code for different paths and we use a router package so that we don't have to determine which path the user is on on our own.

**react-router** contains the logic you could say but to be able to render something to the dom, so to tell react to render something, we need **react-router-dom** too,

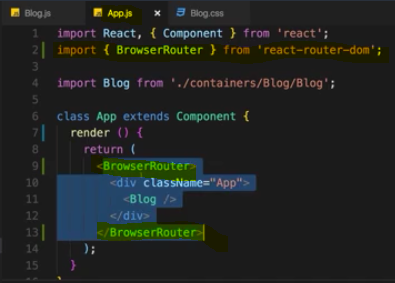


**we first of all need to enable routing in our react app.**

**we do this in the index.js or the app.js file.**

**There you now need to wrap the part of your app which should be able to render routes and to read routes, you need to wrap it with a component you import from the react-router-dom package.**

**this BrowserRouter object. Now you should wrap everything in your app which should be able to use routing with that.**



We installed both react-router  and react-router-dom . **Technically, only react-router-dom  is required for web development. It wraps react-router  and therefore uses it as a dependency.**

**We don't need to install react-router  on our own for it to work. You can omit this installation step**

Now import



**Here you define the path for which this route should become active then you need to define what should happen when this is the active path, So when react router finds out that this is the active path.**

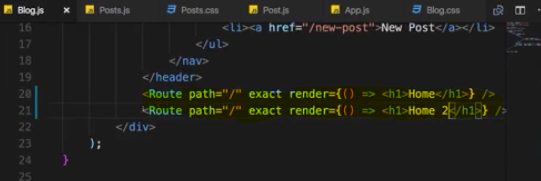




**You use the path here to tell react router does my path start with this? Except for the case that you define exact, then the question would be is my complete path like this?**

**And then here render determines what the react router should render to the screen in this component's place here, so route simply replaces itself with the Content you define in render.**

**you can of course use multiple routes, even for the same path.**





we render both home and home 2 beneath each other.

**so if path is the current path or if you don't specify exact, if it is the prefix of the current path and then render this content.**

**This is the core functionality of the router** .

## **Rendering Components for Routes**

**We can render components for routes in two ways**

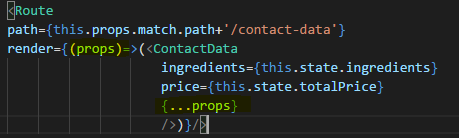
**1 : we can use render props incase we need to pass some props too**



**But in this case we don’t get Route props like history , match and so on…**

**due to the way we are loading Component by basically rendering it manually down here with the render method, we don't have the history object available in there.**

**So to get these props in render component we need to manually pass these props we get in render method into rendered component**



**Alternative way is -**

we can wrap the ContactData component with this withRouter helper method.

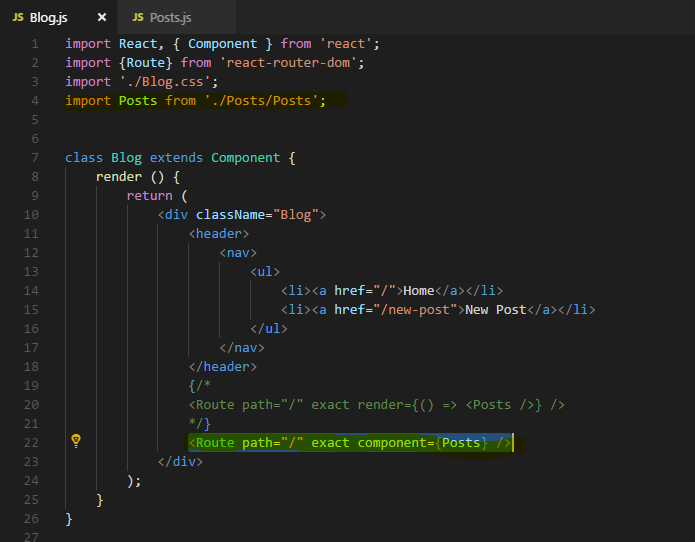
**Export default withRouter(Posts); - here Posts is parent component**

**2 : by component property(as a props )**

<Route path="/" exact component={Posts} />

this allows us to pass a component which should be rendered in this route's place.

Now component is not a string so we don't write “Posts” or {<Posts />} here, that wouldn't work, component needs to be a reference to the function or class we want to use.



**So this now shows us how we can load components just like with render but with the component property.**

**Obviously this is the use case you're probably going to use a bit more, render is most important for short info messages.**

**You shouldn't really render a whole component's code here even though you could technically do that but that will mess up your code, will make it very hard to maintain but loading components like this here that is the default case which will use a lot.**

**Now we should not use <a href….> here because it reloads the page**

**we need to change to behaviour so that we don't have a normal link here which reloads the page but that instead we prevent the rebuilding of the page and let react router only re-render parts of the dom or tell react what to re-render to be precise that needs to be re-rendered.**

**For that we need to change our links, right now we use normal anchor tags and this is not what we really use when using react router, there are we have a special component we can use, the Link component. As the name suggests, link allows create a link.**



**We use this instead of the anchor tag,**

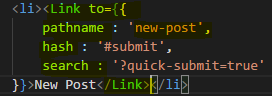
**Link alone of course doesn't do much because we need to tell react router where**

**this link should lead to.**



**Now I said in its simplest form, to is a slash(any string), to can also be a more complex element, it can be a javascript object and hence needs to be wrapped in single curly braces to output dynamic content and then the dynamic content is a javascript object, so one more pair of curly braces.**

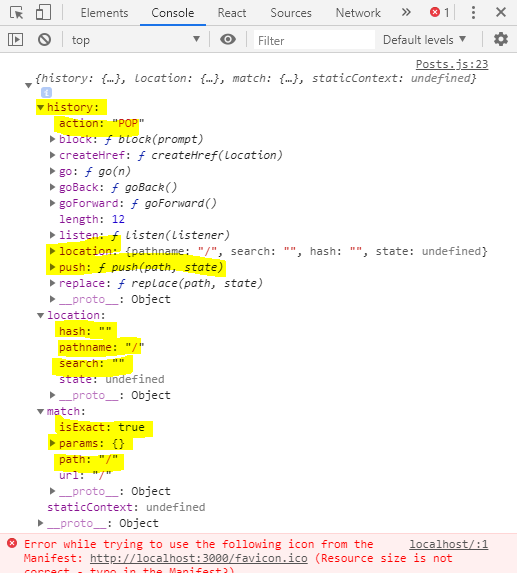
**And there, we can now configure where we want to go to,**



**search allows us to add queryParams like quick submit equals True**

## Routing-related Props

**react router gives us some extra information about the loaded route through props.**



**this is the props object and history, location and match, this is not something we set up, this is passed by react router.**

**The history object also has some methods we can use like push to interact with that and push a new page programmatically without needing a link component,**

## **“withRouter” HOC and Route Props**

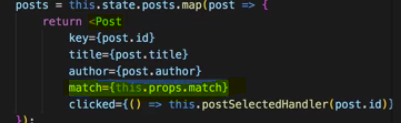
**what if we actually want to get this information not in one of our containers, so not in a component which was loaded through a route but in a component which is rendered as part of such a container.**

Now if you wanted to use the props down to component chain, there are **two ways to get access**,

**1 : one of course is we can pass them on with the spread operator and this props, this passes any props this post container has onto the post component.**



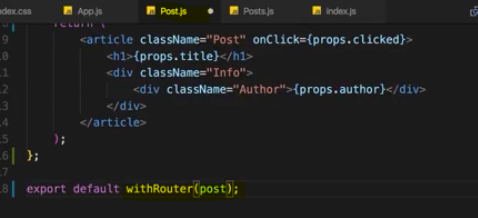
we could also explicitly target certain props like for example if you want to pass on the matched property, We could also do this.props.match. Now this is one way of passing it on

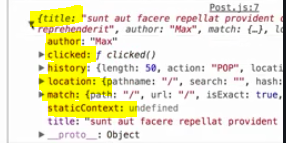


**2 : there also is a higher order component we can use on the post component, there we can import something which is called withRouter from react-router-dom.**

**Now withRouter is a higher order component which we use by wrapping our export with it**





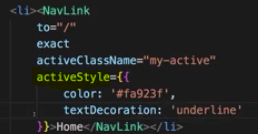
Output - 

**So withRouter adds these props to other components, to any component we wrap with it.**

**NavLink**

**It's pretty similar to link but it has some extra props which allow us to define some styling for the active link.**

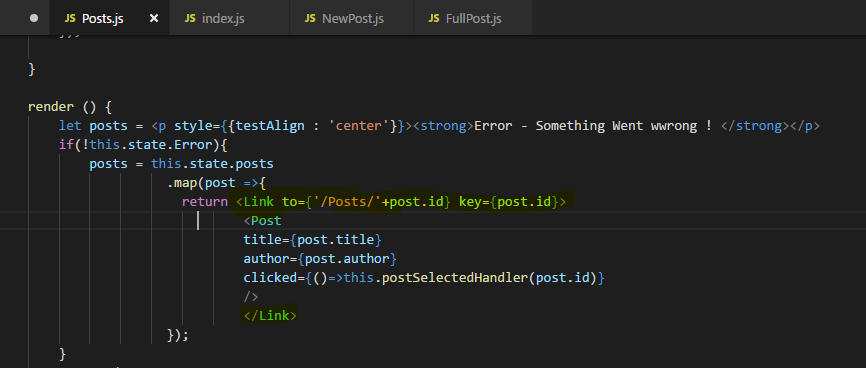


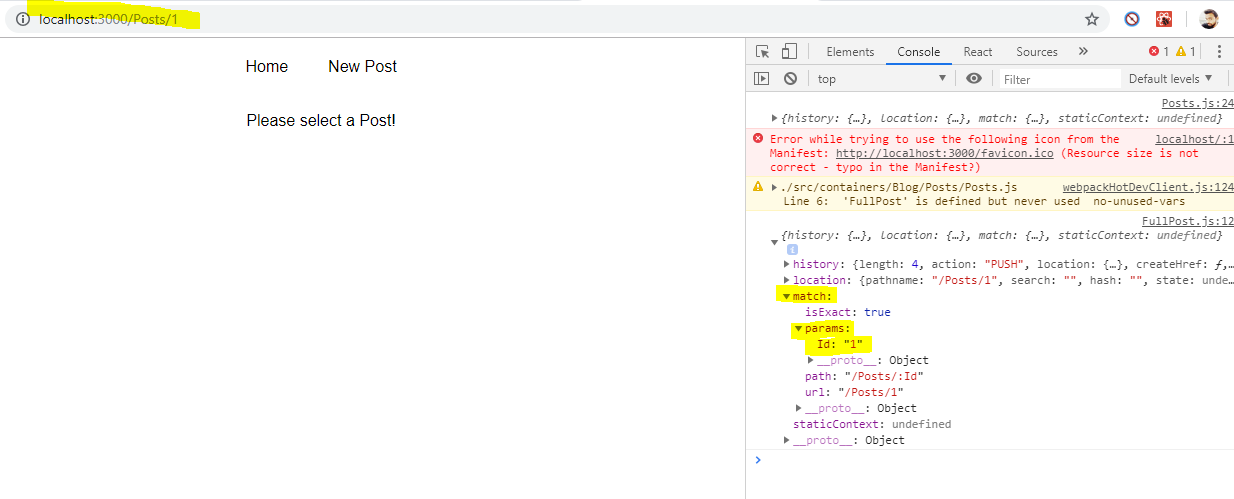


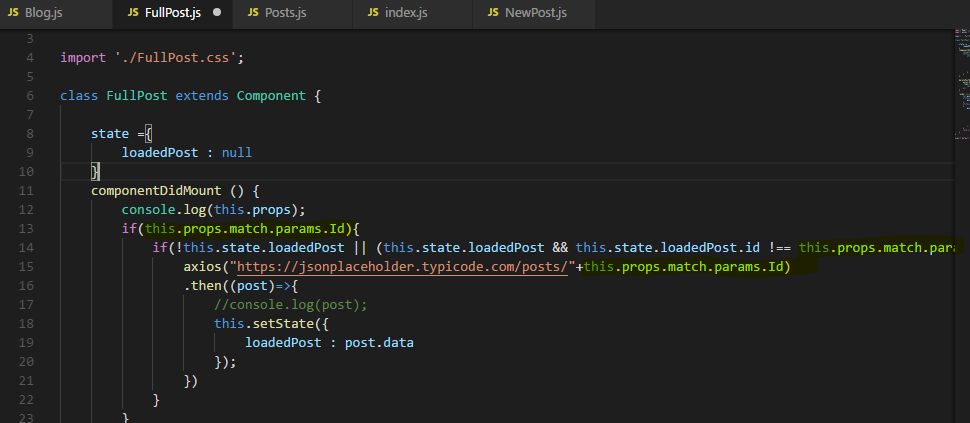
## Route Parameters : Passing route Parameter

**For this, when defining the route, we need to inform react router about the fact that we now have a dynamic element in this URL.**





Getting the route parameters



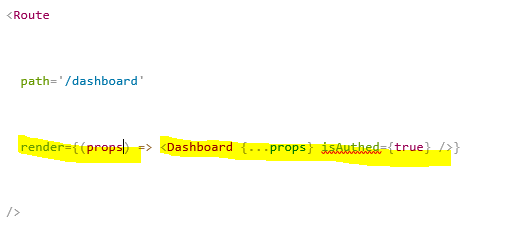
Any query parameter we can get in - props.location.search

## **Pass props to React Router's Link component**

there are two ways to pass data from a Link through to the new route: URL parameters and state. URL parameters work great for strings, but break down after that. By making the Links to prop an object, you can pass along any sort of data you need under the state property and that data can be accessed in the new route under props.location.state.



## **Pass props to a component rendered by React Router**



## **Switch**

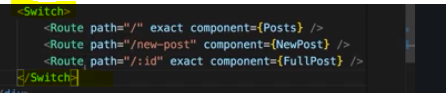
**Remember that all routes here are rendered if they match the path.**

We can still tell react router to load only one of all these routes at a time, by wrapping our route config here with another component provided by the react-router-dom package, this **Switch** component.

**Switch tells the react router, hey please only load one of the routes. The first one actually you find that matches from a given set of routes.**

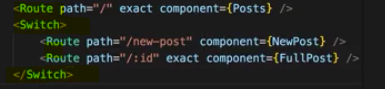
So now we can wrap our three routes with switch and the first route that matches a given path will be loaded and thereafter, it will just stop analyzing the routes, it won't render any other route.





**of course you can also mix it,**

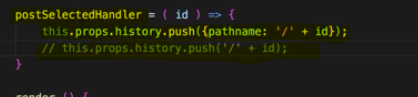
**you can put one route outside of switch. So this will always get analyzed at least and potentially also rendered if it fits and then one of the following**



**Navigating between Routes Programmatically**

**you also sometimes have use cases where you want to navigate after something finished, after a HTTP request was sent**

**Now to do this, we can take advantage of that history object we received on the props**



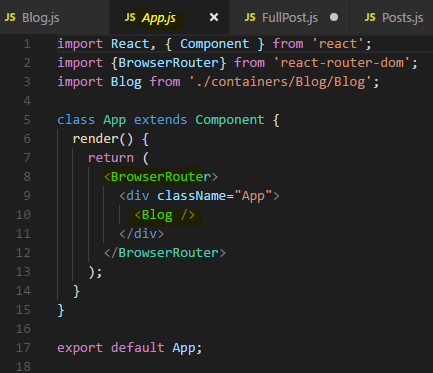
**history prop also has a go back method.This simply goes back to the last page,**

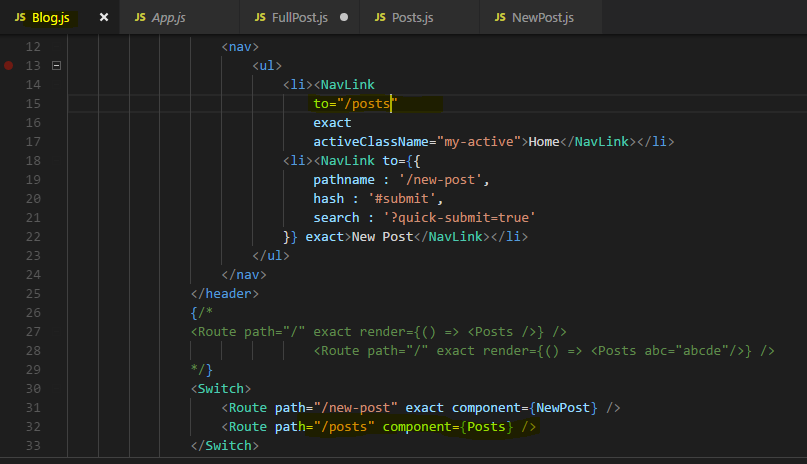


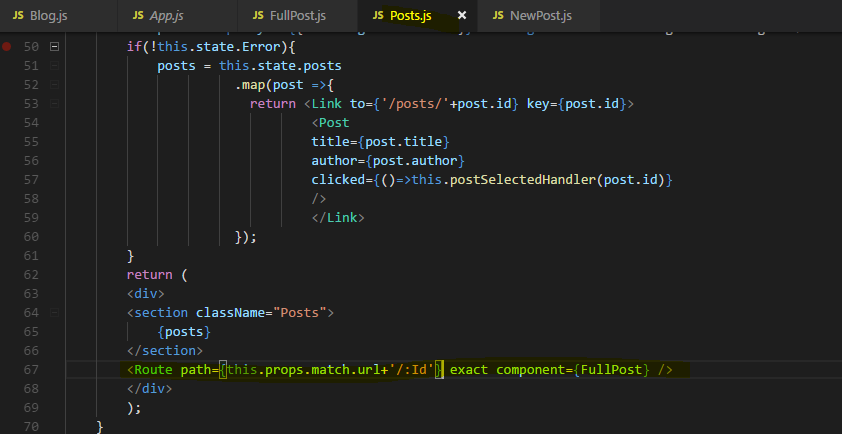
**Nested Routes**

sometimes you also have a set up where you want to create a nested route. So **where you want to load a certain component or where you want to render certain content inside of another component which is also loaded via routing.**

**You can use the route component where ever you want in your application as long as the page the component where you are using it is wrapped by that BrowserRouter**





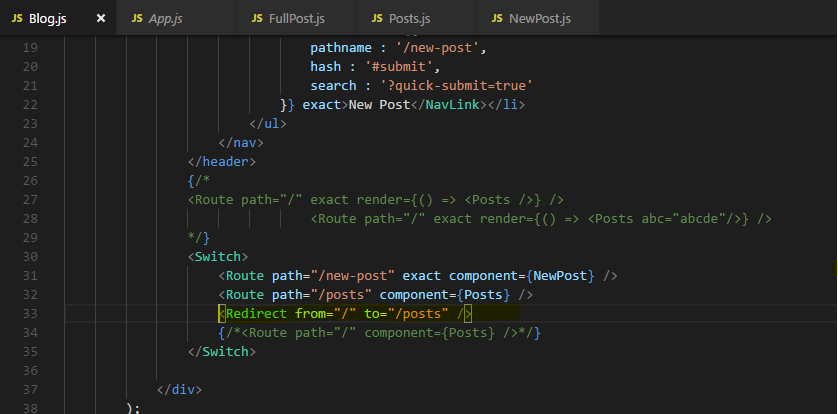


**this.props.match this match object and in there there is a url property which is basically the url we loaded thus far.**

**Redirect Request to different Route(Redirect Component)**

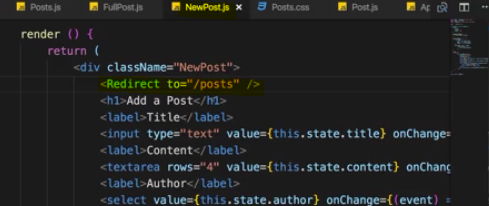
The **Redirect** component as it is a component is simply used in your jsx code, there you can add it to your switch statement and then you specify a **from** property, so from which route you want to navigate to which. For example, from / **to** /posts and then it's a self-closing component.

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



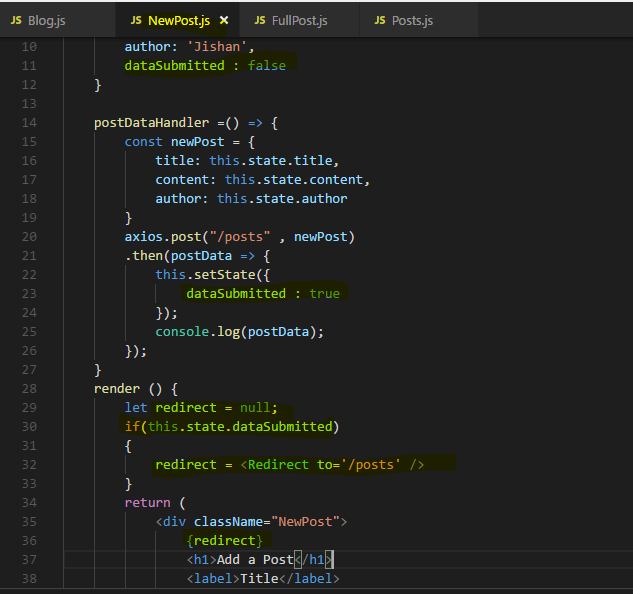
**So now we have that redirect statement in the switch statement and it's just like another route, the only difference is it doesn't render content, it simply just changes the URL so that we then reach another route for which we render content.**

**if we place redirect in jsx outside of a switch statement, we always have to redirect with the to property though, we can't use from.**



**The issue with that is if we click onto new post we are immediately redirected back to posts because the redirect component is rendered and when whenever this is rendered, it redirects us, so we have no chance of entering content.**

**Therefore we need to render this conditionally and since it's a normal component as all of js react router is, we just have to render it conditionally as we render any other content conditionally.**



**Using the History props for Redirecting ( Replace method)**

**history** prop has a couple of convenience methods we can use for replacing the current route or pushing a new route.

**This.props.history.replace(‘/posts’);**

Or

**This.props.history.replace({pathname:’/posts’});**

**Technically push pushes this page onto the stack, so if we click the back button, we go back to the new post page. Whereas redirect replaces current page so if we then click the back**

**after redirecting and not by pushing, you can go back but you won't go back to the last page because redirect replaces the current page on a stack, it doesn't push a new one.**

**Handling 404 case (Unknown Routes )**



**So here I'm going to output Page not Found , of course you could again just set it, use component and render 404 error page component you created , but this route will catch any routes which are not handled prior to it.**

**So any unknown routes in our case and it will output this.**

## **Other Interview Questions**

