react-router

Routing in react application

Our goals

- Combine routing with react
- Understanding best practices regarding routing
- Dealing with params
- Navigating to different routes
- Lazy loading

Routing job in WWW

- Mark the page we are seeing
- UX - look at the url and know what the page is about
- SEO

Routing in SPA

- Although single page application is run and managed by JS we still require the url to change just as regular web pages do
- Navigating to a different page should show a different page, and reloading will bring us back to the same page
- The transition to different route can be done via JS without the need to refresh the entire page
- This means transitions will resemble a desktop app and will be super fast and user friendly

Static routing VS Dynamic routing

- Static routing is defined in a configuration before the framework is initialised for example
  - Angular
  - Express
- Dynamic routing is determined during runtime
- Dynamic can embed conditional routing content in the middle of our code
- For example in dynamic routing it is easier to do things like

```
if (clientScreenSize === Mobile) {
    render in this path the mobile home
} else {
    render in this path the regular homepage
}
```

Routing in React

- What we want with routing in React
  - Use HTML5 history to change the location without reloading
  - Support different platforms
  - As simple as possible and using the react approach to solving problems

react-router

- react-router is a package that provides us the following
  - choose a router
  - conditionally display component based on the URL
  - navigate to different pages

Router

- The router answer the question: How will the routing be done?
  - will we use HTML5 history?
  - Will we use hash routing?
  - Will the routing be static?
  - Will the routing be in memory?
- Our first step will be choosing the router and in modern web application the choice will be to use HTML5 history which is BrowserRouter
- The Router is provided to all the subtree (what pattern is this?) so we will need to wrap our subtree with the router of our choice

BrowserRouter - EX

- BrowserRouter is a context provider of the router which is based around HTML5 history
- It will provide the context to all the subtree
- Let’s start by wrapping our root component with the BrowserRouter

Route - EX

- Route is a component used to conditionally display a react component if the route match
- It will accept the following properties
  - path - string/string[] of all the path that considers match, the format is string-to-regexp
  - component - the component to render when there is a match
  - render
  - children
  - exact - a match should be exact the default is that prefix match as well
  - Create an about page that will show in the /about route

Router props

- When a component is rendered with the router it will recieve the following in the props
  - history
  - location
  - match

route params

- URL can also be used to transfer data to the next page
- What are the ways we can transfer data?
- What is best practices and how do I choose which way?

Params - EX

- Try to pass route params to the about page we created before
- Try to pass query params as well
- Make sure you print them in the component

Switch - EX

- Using Switch we can group multiple Route and it will try everyone until the first match
- It can be used to solve a 404 page
- Wrap the Route we created before and if no match display a 404 page

Redirect - EX

- Component used to navigate the user to a different page
- Add a Redirect to our Switch and if we are in the path /admin we should direct the user to the about page
- We can also use the history passed through the props to navigate with code

Link - EX

- To link to different pages we cannot use the anchor tag (why not?)
- We can link to different pages using the Link component
- Create a navigation bar for our app

Lazy loading

- We need to keep our app size as small as possible
- The smaller our app files will be, the faster our app will load and run
- our app should should also scale, the more pages we add does not necessarily means bigger bundle size
- Some pages we do not need to eagerly load when our app starts
- for example the settings page the user will not enter every time he goes in our app, it makes more sense to only load it when the user visits our site.
- Create a Link to a settings page and let’s try to lazy load that page

Dynamic import - EX

- Dynamic import let’s us load files dynamically in our code
- it accepts a path for the js file and it will load it with ajax and return a promise which will be resolved with the fil
- We will use this feature to lazy load our settings component
- Since this feature is still experimental we need to add the proper plugin to babel in order to support it
- Create a .babelrc file and install the plugin:
  - @babel/plugin-syntax-dynamic-import
  - In the .babelrc file add plugins with the plugin you installed

Dynamic import - EX

- Dynamic import let’s us load files dynamically in our code
- it accepts a path for the js file and it will load it with ajax and return a promise which will be resolved with the fil
- We will use this feature to lazy load our settings component
- Since this feature is still experimental we need to add the proper plugin to babel in order to support it but if using create-react-app it should already be enabled
- Create a .babelrc file and install the plugin:
  - @babel/plugin-syntax-dynamic-import
  - In the .babelrc file add plugins with the plugin you installed

React.lazy

- **Let’s you render dynamic import as regular component**
- **It will get a function that will return a promise**
- **That promise should be the file with the react component we want to load set as the default export**

React.Suspense - EX

- While our component is loading we need to show some default content we do this with the Suspense component
- We specify the fallback property with the component we want to show while our component is loading
- Finish adding lazy loading for our settings page

Summary

- From all the popular frameworks out there and their routing, react routing is probably the easiest to learn
- Dynamic routing is powerful and easy to learn
- React router is a super simple library and not too many components to learn there
- Don’t forget to use lazy loading to reduce your app size