React router

URL job in web apps

- request resource from server
- SEO
- Save current page we are in
- transfer data with query params
- Save state

history api

- When requesting a new url the browser will reload a new page
- In SPA we want to keep the user on the same page and not reload the page
- We still want the URL to change even in SPA
- with the history api in html5 we can now change the url without reloading
  - **history.pushState({}, "stam", '/wat/') - the first object will be sent to the pop state event**
  - **history.replaceState({}, "stam", '/wat/')**
- With React we wont use the **history** api directly rather we will use a package called React Router

What is React Router

- react router is a react component
- you install it with **npm: npm install react-router-dom --save**
- you have to decide which Router to use base on the history api
  - BrowserRouter
  - HashRouter
  - MemoryRouter
- The router expects to see a single child
  - for this we will have to wrap our app in a global component
- Let’s try and install the router and create an app component displaying hello world message

Route

- route will render a react component based on path match
- passing exact prop will match if the entire path match
- the component prop will decide which component to render based on the path
- wrapping in Switch will only render the first match
- for your app to support reload you need to configure the server to serve the app on every url your router can transfer to
- Let’s create two pages:
  - about page
  - home page
- When we have a path match, the component is passed with a match object
- We will use HashRouter so we can train without a proper server

Catch all route

- you can specify a route without a path
- that route will always match
- you can use the catch all route to display a 404 page

Navigation bar - Link

- if we create navigation with **<a>** tags the entire page will be reloaded
- **Link** is rendered to **<a>**
- the **Router** is passed to **Link** and it does the transition with the **Router**
- clicking the **Link** will cause the path to change without the browser reloading the page
- **Link** is passed with **to** property that tells where to transition
- Let’s create a navigation bar to move from the home page to the about page

Navigate by code

- the router is passing extra items in the props
  - location - data about the path
  - history - can be used to interact with the history
- you can use **history.push**

Nested Routes

- currently we have a global app component where we can put common things to every route
- it’s common that we want to put common things for some routes not all of them
- it’s common to see this behaviour when we have a list view and a details view
- Let’s create another route called **/questions/** which will display a common header and nested children for that route
- you have to play with the exact so the parent will be rendered and so will the child

url params - query params

- url is also used to pass params
- react router matches a url by pathname, query params are not considered in the matching
- query params syntax: <protocol>://<host>/<pathname>**?param1=value1&param2=value2**
- query params are usually used when we have same page that change according to dynamic param
    - search page
    - filters on map page
- react router will pass the **search** in **this.props.location.search**
- react router doesn’t deal with query params parsing, we can install another library to deal with that

let’s create a link that transfers query params to the questions page

url params - matrix params

- syntax:
  - **/tasks/:id/**
- you can grab the param with: **this.props.match.params**
- let’s try and pass a param to the the questions page

React Router with Redux

- For our app state library we would like to use Redux
- we might want to render some components inside a page
  - for example when a query param is changing
- There is a package that binds the route to the redux state
- the package is called: **react-router-redux** and can be installed with npm