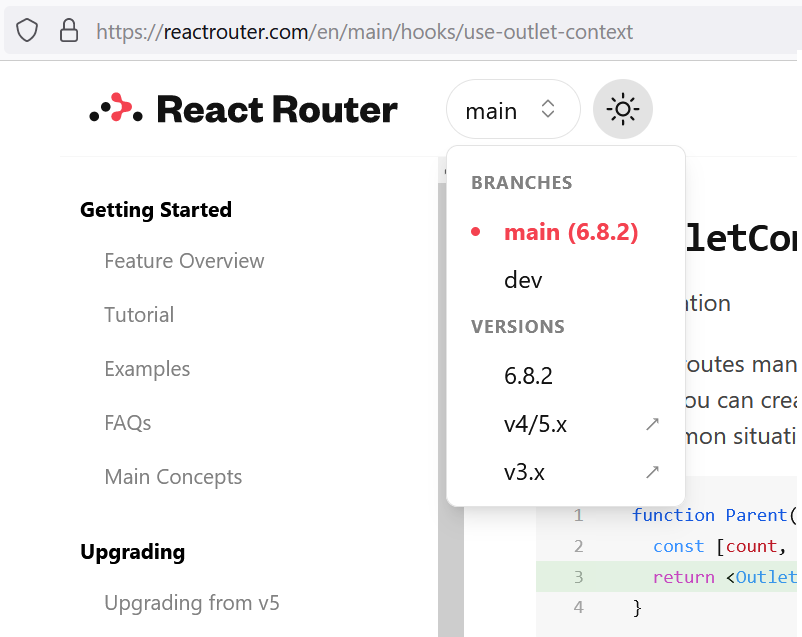
**React router 2023K**

It at least used to be a problem to get to wrong version of the documentation easily, if you google for react-router stuff. Possibly now better, but still check that you are on correct version of the documentation if something sounds strange:

****

* Different react-router-dom versions! E.g. 4 -> 5 -> **6**

**use the React Developer Tools**

React Developer Tools (in dev tools as separate tab for Chrome). Select component to see: Hooks, Matches, History, Location / Key, Props, State

* ((Class: Look at the React Dev tools (History, Location: e.g. query string params , Match: e.g. route params)))
* Hooks: Look at the React Dev tools (History, Location: e.g. query string params , Match: e.g. route params)

Console: $r (contents of the selected React component) $r.render( )

(And similarly if selecting from the DOM with the Elements tab, Console: $0 or e.g. $0.click()

Try also the search feature

**Routing**

* npm i react-router-dom,
* <BrowserRouter> …whole app here… </BrowserRouter>
* <Routes>
  + nowadays React router v6 is intelligently matching to correct route. So order your Routes naturally.
* <Link>/<NavLink> to have SPA plumbing instead of full page reload requests
  + NavLink to show the active link with different styling
* <Rout path="/xyz" element={<SomeElement />} />
* <Route path="\*" … matches with the rest!
* <Redirect from="x" to="y" />
* (MemoryRouter is good for tests without needing Browser)
* (StaticRouter for server-side rendering)
* (NativeRouter for React Native)
* element=<Member**Layout** /> for layout shared by nested routes and inside that layout component
* <Outlet /> for displaying each nested route's own component content.
  + <Outlet **context** = { name: {objProperty:value1, objProperty:value2} }

**Programmatic navigation (Not needed in basic app example)**

* Navigate or useNavigate
* History/History stack: Push, Pop, Replace   
  Business case example where needed: (IdeaList->IdeaDetails->IdeaDelConfirm (+yes)->IdeaList)

Routing introduction video, v6: <https://www.youtube.com/watch?v=Ul3y1LXxzdU>

* Until 16:47 the basics
* until 23:53 "basics" about **nested routes** (shared relative link start, shared layout, shared context)
* until 28:53 some advanced topics about nested routes.
* From 32:30 or so NavLink with isActive
* 36:30 useNavigate hook
* 43:17 state={}, useLocation, location object w. e.g. "search", "hash", and "state"

**Main concepts** - <https://reactrouter.com/en/main/start/concepts> Read at least these

* Location
* History Stack, History
* Client Side Routing
* Route
* Nested Routes
* Relative Links
* Outlet
* Layout Routes (Special case of using your own Layout component)

BrowserRouter - <https://reactrouter.com/en/main/router-components/browser-router>

Routes - <https://reactrouter.com/en/main/components/routes>

Route - <https://reactrouter.com/en/main/route/route> Url, \*, :param,

Link - <https://reactrouter.com/en/main/components/link>

NavLink - <https://reactrouter.com/en/main/components/nav-link>

useParams( ) - <https://reactrouter.com/en/main/hooks/use-params> /:id

useSearchParams( ) - <https://reactrouter.com/en/main/hooks/use-search-params> ?name=Joe&loc=Turku

(Layout was just an element of the parent route, element={<MemberLayout />} )

Outlet - <https://reactrouter.com/en/main/components/outlet> <Outlet **context**={{}}/>. parent

.useOutletContext( ) - <https://reactrouter.com/en/main/hooks/use-outlet-context> child

**Programmatic navigation**

Navigate - <https://reactrouter.com/en/main/components/navigate>

useNavigate - <https://reactrouter.com/en/main/hooks/use-navigate>

**Let's look at the case for example!**

* react-router-dom version? **6** (package.json)
* routing architecture/ navigational model? (idea-demo-short or current case)
* Are routing contexts used so far? (Are other React contexts used?) (Redux is not used in this project)
* What of the **Routing features** learned today would **you take into use** while this semester case is expanding?
  1. Case has e.g. multiple Users, but user management (with roles and rights and ownerships?) is just one corner of the case. Thus, many views about users/roles/ownerships.
  2. You want the user to associate **similar looks** of certain views **to certain related activity** so that the navigation inside the app is intuitive.
  3. You also want the user to be able to find the: **last** read/handled/modified/searched **items** instead of always starting from all 1000-3000 items.
  4. You want to know what is the status of the app, meaning what is going on right now, kind of providing the user with the feeling of **statefulness**.
  5. Even if we did not see much of the programmatical routing / navigation, do you see where it could be used? (History stack and its manipulation: push, pop, replace)

Suggestions:

1. Creating separate "corners" or "areas" in the app, with nested routes and their layouts. E.g. member management, buildings spaces and equipment, departments and curricula, teachings and their equipment requirements. And calculation versions and results.
2. Each nested corner's layout could have something reminding of it. E.g. icons or pics of people, buildings, instruments. And possibly different background color or accent color?
3. Context could be used to save lists of the recent items?
4. Keeping in some context the currently handled calculation round? Currently logged in user (of course token based check for that)?
5. IdeaList, (select one) -> IdeaDetails (press Delete) -> IdeaDeleteConfirmation (click Yes)-> (notification about deletion) -> IdeaList   
   *=> Replace the non-wanted back history with the IdeaList. We do not want to go to the IdeaDetails for Idea that was already deleted*.