06-Short-Link-App

Logging Out and Tracking Auth Status

- We are going to wire up the log out button
- first go to Link.js
- remove line of code in the onLogout() function
- delete the browserHistory import
- we are going to be importing Account from accounts-base

```
import React from 'react';
import { Accounts } from 'meteor/accounts-base'



export default class Link extends React.Component{
    onLogout(){


    }
```

- now all we do call it inside our onLogout(){}

```
import React from 'react';
import { Accounts } from 'meteor/accounts-base'



export default class Link extends React.Component{
    onLogout(){
        Accounts.logout();
    }
    render(){
        return (
            <div>
                <h1>Your Links</h1>
                <button onClick={this.onLogout.bind(this)}>Logout</button>
            </div>
        );
```

```
    }
}
```

- head over to the browser and check on the console and check if you are logged in first
- if it says null, then you are logged out

```
require('meteor/meteor').Meteor.user()
```

- no login to your account in the browser
- now go to your /links url
- and click logout
- now if you type

```
require('meteor/meteor').Meteor.user()

null
```

- now we can begin to plan what to do when someone logs in or logs out
- we are going to be using tracker autorun to get that done
- and that will happen in our main.js for the moment
- lets now import the named export { Tracker }

```
import { Meteor } from 'meteor/meteor';
import React from 'react';
import ReactDom from 'react-dom';
import { Router, Route, browserHistory} from 'react-router';
import { Tracker } from 'meteor/tracker';

import Signup from '../imports/ui/Signup';
import Link from '../imports/ui/Link';
import NotFound from '../imports/ui/NotFound';
import Login from '../imports/ui/Login'

window.browserHistory = browserHistory;
```

- now lets use that Tracker with the autorun() method down below
- this method only takes on argument, a function
- then we will created a const variable and assign it to the Meteor.user()
- We then are going to make that value which is null to a boolean, and we do that by adding !!
- so now null will become true and if it's not null it will be false

```
const routes = (
  <Router history={browserHistory}>
    <Route path="/" component = {Login}/>
    <Route  path="/signup" component={Signup}/>
    <Route path="/links" component={Link}/>
    <Route path="*" component={NotFound}/>
  </Router>
);

Tracker.autorun(()=>{
  const isAuthenticated = !!Meteor.userId();
});

Meteor.startup(()=>{
  ReactDom.render(routes,  document.getElementById('app'));
});
```

- lets play with this by console.log

```
Tracker.autorun(()=>{
  const isAuthenticated = !!Meteor.userId();
  console.log('isAuthenticated ',isAuthenticated);
});

Meteor.startup(()=>{
  ReactDom.render(routes,  document.getElementById('app'));
});
```

- now head over to the browser and should see in the console

```
isAuthenticated  false
```

- now lets log in to change that
  - and you should see in the console change to true

```
isAuthenticated  true
```

- now in the browser go to you /links

- and click log out, now should see

```
isAuthenticated  false
```

- now we are going to keep track of the users page, and we get that information from the browserHistory
- lets play with that in our console, all you have to do in the main.js is...

```
window.browserHistory = browserHistory;


Tracker.autorun(()=>{
  const isAuthenticated = !!Meteor.userId();
  console.log('isAuthenticated ',isAuthenticated);
});
```

- now over the console
- what matter here is the pathname

```
browserHistory.getCurrentLocation()
{pathname: "/links", search: "", hash: "", state: undefined, action: "POP", …}
```

- now that we now how to use getCurrentLocation head over to main.js and remove that line of code
- and we can use that in the Meteor.autorun
- create a const variable pathname
- and give the getCurrentLocation() value
- that returns an object, so we want to target pathname

```
Tracker.autorun(()=>{
  const isAuthenticated = !!Meteor.userId();
  console.log('isAuthenticated ',isAuthenticated);
  const pathname = browserHistory.getCurrentLocation().pathname;
});
```

- we next are going to define what pages should serve what purposes and this is going to happend via 2 arrays
- so back in our main.js lets create this
- a variable called unauthenticatedPages, this will include all of the pages that a user should not be able to visit if they authenticated-which are the login page, and the signup page
- the next array is authenticatedPages, and this is when you are logged in

```
const unauthenticatedPages = ['/', '/signup'];
const authenticatedPages = ['/links'];


const routes = (
  <Router history={browserHistory}>
    <Route path="/" component = {Login}/>
    <Route  path="/signup" component={Signup}/>
    <Route path="/links" component={Link}/>
    <Route path="*" component={NotFound}/>
  </Router>
);
```

- now we can use their values and their path name to determine what type of page a user is on
- now we are going to create two more constants down in our Tracker.autorun

```
const routes = (
  <Router history={browserHistory}>
    <Route path="/" component = {Login}/>
    <Route  path="/signup" component={Signup}/>
    <Route path="/links" component={Link}/>
    <Route path="*" component={NotFound}/>
  </Router>
);



Tracker.autorun(()=>{
  const isAuthenticated = !!Meteor.userId();
  const pathname = browserHistory.getCurrentLocation().pathname;
  const isUnauthenticatedPage = unauthenticatedPages.includes(pathname);
  const isAuthenticatedPage = authenticatedPages.includes(pathname);
  console.log('isAuthenticated ',isAuthenticated);
```

- here we used the include() method which is an array method and we are passing the pathname variable above to check if the path name is '/' or 'links' and so fort

- now we are going to use if statements when certain conditions are met or not met

```
Tracker.autorun(()=>{
  const isAuthenticated = !!Meteor.userId();
  const pathname = browserHistory.getCurrentLocation().pathname;
  const isUnauthenticatedPage = unauthenticatedPages.includes(pathname);
  const isAuthenticatedPage = authenticatedPages.includes(pathname);

  if(isUnauthenticatedPage && isAuthenticated ){
    browserHistory.push('/links');
  }
  else if(isAuthenticatedPage && !isAuthenticated){
    browserHistory.push('/');
  }

});

Meteor.startup(()=>{
  ReactDom.render(routes,  document.getElementById('app'));
});
```

- now go to your browser and you will notice that isAuthenticted should be false
- if we try to go the /links url it will not let us, it will redirect us to the root of the webapp
- now lets log in to test this by typing our email and password
- and it does work, it takes us to the /links page
- now lets try to redirect to the root of the app by typing in the url locahost:3000/
- and that won't let us do it until we logout

Private and Public Routes

- we are going to be fixing a problem when the user clicks the back button, right now when the user clicks the back button it takes to a page where it's supposed to be private so we are going to be fixing some of the routers, so back in our main.js
- we are going to define two functions on for the public route and another for the private
- theyre going to be passed on as props, we are first going to be working on the public pages which are Login '/' and Signup '/signup'

```
import Signup from '../imports/ui/Signup';
import Link from '../imports/ui/Link';
import NotFound from '../imports/ui/NotFound';
import Login from '../imports/ui/Login'
```

```
const unauthenticatedPages = ['/', '/signup'];
const authenticatedPages = ['/links'];

const routes = (
  <Router history={browserHistory}>
    <Route path="/" component = {Login} onEnter = {}/>
    <Route  path="/signup" component={Signup} onEnter={}/>
    <Route path="/links" component={Link}/>
    <Route path="*" component={NotFound}/>
  </Router>
);
```

- we then are going to create the functions up above

```
const unauthenticatedPages = ['/', '/signup'];
const authenticatedPages = ['/links'];
const onEnterPublicPage= ()=>{

}
const routes = (
  <Router history={browserHistory}>
    <Route path="/" component = {Login} onEnter = {}/>
    <Route  path="/signup" component={Signup} onEnter={}/>
    <Route path="/links" component={Link}/>
    <Route path="*" component={NotFound}/>
  </Router>
);
```

- now we are going to check if the user is logged in
- is there is a userId it will return a truthy value, if they are we are going to send them to right location
- for some one that is logged in we are going to send them to /links

```
const unauthenticatedPages = ['/', '/signup'];
const authenticatedPages = ['/links'];
const onEnterPublicPage= ()=>{
  if(Meteor.userId()){
```

```
    browserHistory.push('/links');
  }
}
const routes = (
  <Router history={browserHistory}>
    <Route path="/" component = {Login} onEnter = {}/>
    <Route  path="/signup" component={Signup} onEnter={}/>
    <Route path="/links" component={Link}/>
    <Route path="*" component={NotFound}/>
  </Router>
);
```

- now all we have to do is is reference that method to our component down below

```
const unauthenticatedPages = ['/', '/signup'];
const authenticatedPages = ['/links'];
const onEnterPublicPage= ()=>{
  if(Meteor.userId()){
    browserHistory.push('/links');
  }
}
const routes = (
  <Router history={browserHistory}>
    <Route path="/" component = {Login} onEnter = {onEnterPublicPage}/>
    <Route  path="/signup" component={Signup} onEnter={onEnterPublicPage}/>
    <Route path="/links" component={Link}/>
    <Route path="*" component={NotFound}/>
  </Router>
);
```

- now lets test it on the browser, log in first.
- when you're logged in, when clicking the back button it work, when you logout, it redirects you to login page
- and now we are going to have another function that is going to check if we are logged in, if not it redirect us to the root which is the login page

```
const unauthenticatedPages = ['/', '/signup'];
const authenticatedPages = ['/links'];
const onEnterPublicPage= ()=>{
  if(Meteor.userId()){
    browserHistory.push('/links');
  }
}

const onEnterPrivatePage = ()=>{
  if(!Meteor.userId()){
    browserHistory.push('/');
  }
}
const routes = (
  <Router history={browserHistory}>
    <Route path="/" component = {Login} onEnter = {onEnterPublicPage}/>
    <Route  path="/signup" component={Signup} onEnter={onEnterPublicPage}/>
    <Route path="/links" component={Link} onEnter={onEnterPrivatePage}/>
    <Route path="*" component={NotFound}/>
  </Router>
);
```

- the only problem with this whole thing is that we broken the back button
- so we are going to be fixing this
- right now our method is to .push our history, we are adding, what we want to do is replace history
- so we are going to be using the replace() method instead
- so we are going to be making these changes in four different areas in main.js

```
const unauthenticatedPages = ['/', '/signup'];
const authenticatedPages = ['/links'];
const onEnterPublicPage= ()=>{
  if(Meteor.userId()){
    browserHistory.replace('/links');
  }
}

const onEnterPrivatePage = ()=>{
  if(!Meteor.userId()){
```

```
    browserHistory.replace('/');
  }
}
const routes = (
  <Router history={browserHistory}>
    <Route path="/" component = {Login} onEnter = {onEnterPublicPage}/>
    <Route  path="/signup" component={Signup} onEnter={onEnterPublicPage}/>
    <Route path="/links" component={Link} onEnter={onEnterPrivatePage}/>
    <Route path="*" component={NotFound}/>
  </Router>
);



Tracker.autorun(()=>{
  const isAuthenticated = !!Meteor.userId();
  const pathname = browserHistory.getCurrentLocation().pathname;
  const isUnauthenticatedPage = unauthenticatedPages.includes(pathname);
  const isAuthenticatedPage = authenticatedPages.includes(pathname);

  if(isUnauthenticatedPage && isAuthenticated ){
    browserHistory.replace('/links');
  }
  else if(isAuthenticatedPage && !isAuthenticated){
    browserHistory.replace('/');
  }
  console.log('isAuthenticated ',isAuthenticated);

});

Meteor.startup(()=>{
  ReactDom.render(routes,  document.getElementById('app'));
});
```

- to check if this works follow these steps
  - log in
  - open a new tab
  - go to google.com or some other random website

- now go to localhost:3000
- it should take you the links page
- click back
- it should take you back to google page
-