12-Short-lnk-app

Setting up an insecure System - Part II

- lets create a brand new component, its going to be in the ui folder
- it's going to be called LinksList.js - its going to be resposible for fetching the links information from the database and also for rendering all of the items
- in the LinksList.js we will be import React and define the class and save it

```
import React from 'react';


export default class LinksList extends React.Component{
    render(){
        return (
            <div>
                <p>Links List</p>
            </div>
        );
    }
}
```

- now we are going to render this component in the Link.js component
- go ahead and import that default export
- and now we can export it
- this is what Link.js should look like

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


import { Links } from '../api/links';
import  LinksList  from './LinksList';


export default class Link extends React.Component{
    onLogout(){
        Accounts.logout();
    }
    onSubmit(e){
        const url = this.refs.url.value.trim();
        e.preventDefault();
        if(url){
            Links.insert({url});
            this.refs.url.value = '';

        }
    }
    render(){
        return (
            <div>
                <h1>Your Links</h1>
                <button onClick={this.onLogout.bind(this)}>Logout</button>
                <LinksList/>
                <p>Add Link</p>
                <form onSubmit={this.onSubmit.bind(this)}>
                    <input type="text" ref="url" placeholder="URL"/>
                    <button>Add Link</button>
                </form>
            </div>
        );

    }
}
```

- so what just a put a basic component setup
- nex t what we want to do how to get the links array stored in our database on to the screen
- we are going to be using two life cycle methods
    - ComponentDidMount
        - this component gets called right after the whole LinksList gets shown on the screen
        - and we don't call it, React takes care of that
        - and we here's just an example on the LinksList.js

```
import React from 'react';


export default class LinksList extends React.Component{
    componentDidMount(){
        console.log('ComponentDidMount LinksList');
    }
    render(){
        return (
            <div>
                <p>Links List</p>
            </div>
        );
    }
}
```

-
    - The other life cycle component is called componentWillUnmount and it happens right before a component goes away- again like all built in life cycle components you do not call it, it gets called internally
- here is an example of it in LinksList.js

```
import React from 'react';

export default class LinksList extends React.Component{
    componentDidMount(){
        console.log('ComponentDidMount LinksList');
    }

    componentWillUnmount(){
        console.log('componentWillUnmount LinksList');
    }
    render(){
        return (
            <div>
                <p>Links List</p>
            </div>
        );
    }
}
```

- for the above message to show you need to remove the component, so you do this by login out
- inside of our componentDidMount we will be setting up a Tracker.autorun call, we will be grabbing all of our links out of the miniMongo collection and we are going to render them to the screen
- we also will be using componentWillUnmount to clean up that Tracker.autorun call, we need to clean it other wise your program will be slow
- we are going to be setting up state for LinksList - it is going to be an array and it's going to store of the links that are going to get rendered, which means we are going to need the constructor method that we've used before

- then we'll create a state of objects, and that will include a propery called links, which will be an empty array

```
import React from 'react';


export default class LinksList extends React.Component{


    constructor(props){
        super(props);
        this.state = {
            links : []
        }
    }
    componentDidMount(){
        console.log('ComponentDidMount LinksList');
    }


    componentWillUnmount(){
        console.log('componentWillUnmount LinksList');
    }
    render(){
        return (
            <div>
                <p>Links List</p>
            </div>
        );
    }
}
```

- next lets cut the Tracker.auto run from client/main.js
- and paste it in LinksList inside of componentDidMount

```
import React from 'react';


export default class LinksList extends React.Component{


    constructor(props){
        super(props);
        this.state = {
            links : []
        }
    }
    componentDidMount(){
        console.log('ComponentDidMount LinksList');
        Tracker.autorun(()=>{
            const links = Links.find().fetch()
            console.log('Here are the Links ', links);
          });
    }

```

- now we want to import Tracker and Links
- first over in client/main.js lets delete import Links and copy the Tracker import and paste in LinksList
- now lets import Links in LinksList

```
import React from 'react';


import { Tracker } from 'meteor/tracker';
import { Links } from '../api/links'


export default class LinksList extends React.Component{


    constructor(props){
        super(props);
```

- and now we want to modify the Tracker.autorun in the LinksList.js
- so we want to do is update the component state when we print them to the screen

```
import React from 'react';


import { Tracker } from 'meteor/tracker';
import { Links } from '../api/links'


export default class LinksList extends React.Component{


    constructor(props){
        super(props);
        this.state = {
            links : []
        }
    }
    componentDidMount(){
        console.log('ComponentDidMount LinksList');
        Tracker.autorun(()=>{
            const links = Links.find().fetch()
            this.setState({links});
          });
    }
```

- and they way we are going to render that to the screen is by creating a method
- but first lets call it down below

```
    render(){
        return (
            <div>
                <p>Links List</p>
                <div>
                    {this.renderLinksListItems()}
                </div>
            </div>
        );
    }
}
```

- we then want to create it above

```
import React from 'react';

import { Tracker } from 'meteor/tracker';
import { Links } from '../api/links'

export default class LinksList extends React.Component{

    constructor(props){
        super(props);
        this.state = {
            links : []
        }
    }
    componentDidMount(){
        console.log('ComponentDidMount LinksList');
        Tracker.autorun(()=>{
            const links = Links.find().fetch()
            this.setState({links});
          });
    }

    componentWillUnmount(){
        console.log('componentWillUnmount LinksList');
    }
    renderLinksListItems(){
        return this.state.links.map((link)=>{
            return <p key={link._id}>{link.url}</p>


        })
    }
    render(){
        return (
            <div>
                <p>Links List</p>
                <div>
                    {this.renderLinksListItems()}
                </div>
            </div>
        );
    }
}
```

- now the next thing we want to do is to stop Tracker.autorun inside our componentWillUnmount
- in order to do that we need to store the return value of Tracker.autorun, the way we do that is by...

```
export default class LinksList extends React.Component{


    constructor(props){

        super(props);

        this.state = {

            links : []

        }

    }

    componentDidMount(){

        console.log('ComponentDidMount LinksList');

       this.linksTracker = Tracker.autorun(()=>{

            const links = Links.find().fetch()

            this.setState({links});

         });

    }
```

- and now we can use this.linksTracker on the componentWillUnmount

```
    componentDidMount(){

        console.log('ComponentDidMount LinksList');

       this.linksTracker = Tracker.autorun(()=>{

            const links = Links.find().fetch()

            this.setState({links});

         });

    }


    componentWillUnmount(){

        console.log('componentWillUnmount LinksList');

        this.linksTracker.stop();

    }
```

- this will stop the tracker ever running again
- to check if the autotrack stopped lets headover to the terminal
- and run mongo

```
Last login: Tue Oct 17 05:36:50 on ttys001

joses-MacBook-Pro:short-lnk mendoza$ meteor mongo

MongoDB shell version: 3.2.6

connecting to: 127.0.0.1:3001/meteor

meteor:PRIMARY> db.links.remove({})

WriteResult({ "nRemoved" : 4 })

meteor:PRIMARY>
```

- now go to your browser and check the console, and if all went well you should see nothing changed