05-Short-Link-App

Logging In

logout

- we are going to [illegible] from the form but also wire it to the api
- first grab the constructor function and copy it to the Login.js
- the reason is in case there is no user with that email or if the password doesn't match to the account on file
- also copy the code from the return except the <Link> component from the Signup.js and copy it to the login
- lets make some changes after copied
- and now we copy the handler which is the onSubmit() method from the Signup.js to Login.js
- delete the Accounts.createUser
- so far this is how Login.js should look like

```
import React from 'react';
import { Link } from 'react-router';

export default class Login extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            error: ' '
        };

    }
    onSubmit(e){
        e.preventDefault();

        let email = this.refs.email.value.trim();
        let password = this.refs.password.value.trim();

        // this.setState({
        //     error: 'something went wrong'
        // })
    }
    render(){
        return (
            <div>
```

```
                <h1>Short Lnk</h1>

                {this.state.error ? <p>{this.state.error}</p>: undefined}

                <form onSubmit={this.onSubmit.bind(this)}>
                    <input type="email" ref="email" name="email" placeholder="Email"/>
                    <input type="password" ref="password" name="password"
placeholder="Password"/>
                    <button>Login</button>
                </form>

                login form here

                <Link to="/signup"> Have an account?</Link>
            </div>
        );
    }
}
```

- we are going to create a new method that logs us in, and this method does already exsists and should be on the Meteor Documentation
- we next need to import Meteor to Login.js

```
import React from 'react';
import { Link } from 'react-router';
import { Meteor } from 'meteor/meteor';

export default class Login extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            error: ' '
        };

    }
    onSubmit(e){
        e.preventDefault();
```

```
        let email = this.refs.email.value.trim();
        let password = this.refs.password.value.trim();

        // this.setState({
        //     error: 'something went wrong'
        // })
    }
```

- now down in our onSubmit() method lets call the loginWithPassword method, 1st argument takes an object, where you specify either a user name or email, depending what youre using. in our case it is an email, in our case we will be using the ES6 shorthand, the second argument is the password, and then finally we have our callback function which is going to look identical than the other call back function

```
import React from 'react';
import { Link } from 'react-router';
import { Meteor } from 'meteor/meteor';

export default class Login extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            error: ' '
        };

    }
    onSubmit(e){
        e.preventDefault();

        let email = this.refs.email.value.trim();
        let password = this.refs.password.value.trim();

       Meteor.loginWithPassword({email}, password, (err)=>{
           console.log('Login Callback', err);
       })
    }
```

- now go ahead and create a new account by going to your signup url
- To make sure we are logged in go to the console and type
- and as you can see we do have a new user

```
require('meteor/meteor').Meteor.user()
{_id: "JyLCtt54hk5wxxw3D", emails: Array(1)}
emails
:
[{…}]
_id
:
"JyLCtt54hk5wxxw3D"
__proto__
:
Object
```

- now lets log out by typing

```
require('meteor/accounts-base').Accounts.logout()
undefined
```

- and if we rerun the user account we get a null, which is exactly what we want it

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

- now let's go log in with the same credential we just used
- and we should get Login callback and undefined, the reason we get undefined is because that is the error in our callback in the Meteor.loginWithPassword

```
Login Callback undefined
```

- now if we again the user() method in the console we should see our id

```
require('meteor/meteor').Meteor.user()
{_id: "JyLCtt54hk5wxxw3D", emails: Array(1)}
```

- this process did create a token but we don't manage that token directly, but you can explore it inside the Mongodb in the terminal, go to your Mongo tab, if your don have running just type **meteor mongo**

```
db.users.find()
```

- and this is what we have

```
{ "_id" : "A5pvXqou7S6ykPrhF", "createdAt" : ISODate("2017-10-14T17:36:01.692Z"), "services" : {
 "password" : { "bcrypt" : "$2a$10$Nw.5lolOiptOuEmWRDL9HeWAxUmw48UMHj94OSvJzpVUTymWyjg1O" }, "re
sume" : { "loginTokens" : [ ] } }, "emails" : [ { "address" : "jose@jjmendoza.com", "verified" :
 false } ] }
{ "_id" : "JyLCtt54hk5wxxw3D", "createdAt" : ISODate("2017-10-14T23:36:35.087Z"), "services" : {
 "password" : { "bcrypt" : "$2a$10$euKcgnfKLuZnqUqR5DmCKejv6UJAWDToVw.vO3iQoVMJEYEX5RZv." }, "re
sume" : { "loginTokens" : [ { "when" : ISODate("2017-10-14T23:43:44.248Z"), "hashedToken" : "Djt
idKDgtP6ZgEQRgxsEJWHy+PE9C7+rz7HhIxJnTvE=" } ] } }, "emails" : [ { "address" : "mendozarriff@gma
il.com", "verified" : false } ] }
```


- you should be able to the hashedToken value, and this is our authentication token that gets passed back and fort, and this securily authenticate us with out needing to provide the email and password on every single request
-