128-Testing Login video

- go to Login.js and setup the imports

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

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

    }
```

- next thing we need to do is restructure the exports Login, so our Login is going to be changed and named export instead of default

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

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

    }
```

- and down below we are going to setup that default
- and we want to export the createContainer(), we know that it takes two arguments, the first which is a function, and the second is the component you want to show on the screen, which in our case is Login

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


export 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)=>{
          if(err){
            this.setState({error: 'Unable to login.  Check email and password'});
          }else{
            this.setState({error: ''});
          }
       })
    }
    render(){
        return (
```

```
            <div className = "boxed-view">
                <div className = "boxed-view__box">
                    <h1>Login</h1>

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

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

                    <Link to="/signup">Need an account?</Link>
                </div>
            </div>
        );
    }
}
export default createContainer(()=>{

},Login);
```

- **inside of our function we are going to return an object**

```
export default createContainer(()=>{
    return {

    }
}, Login);
```

- lets create a property loginWithPassword:

```
export default createContainer(()=>{
    return {
        loginWithPassword : Meteor.loginWithPassword
    }
}, Login);
```

- and now we can change the line up above to

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

export 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.props.loginWithPassword({email}, password, (err)=>{
          if(err){
            this.setState({error: 'Unable to login.  Check email and password'});
          }else{
            this.setState({error: ''});
          }
```

```
        })
    }
    render(){
        return (
            <div className = "boxed-view">
                <div className = "boxed-view__box">
                    <h1>Login</h1>

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

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

                    <Link to="/signup">Need an account?</Link>
                </div>
            </div>
        );
    }
}
export default createContainer(()=>{
    return {
        loginWithPassword : Meteor.loginWithPassword
    }
}, Login);
```

- now lets setup the propTypes

```
Login.propTypes = {
```

```
    loginWithPassword: React.PropTypes.func.isRequired
}


export default createContainer(()=>{
    return {
        loginWithPassword : Meteor.loginWithPassword
    }
}, Login);
```

- now check the browser to see if its' working-login and logout
- now lets setup the test cases for Login
- over in the terminal lets quit, and start our test

```
joses-MacBook-Pro:notes mendoza$ npm test
```

- **over in imports/ui create a new file called Login.test.js**
- **copy and paste the same import from PrivateHeader.test.js**

```
import { Meteor } from 'meteor/meteor';
import React from 'react';
import expect from 'expect';
import { mount } from 'enzyme';
```

- now over in Login.test.js lets do you an if statement to chekc if we are in the client side

```
import { Meteor } from 'meteor/meteor';
import React from 'react';
import expect from 'expect';
import { mount } from 'enzyme';

if(Meteor.isClient){


}
```

- **and in that block we are going to have a describe block**

```
if(Meteor.isClient){
```

```
    describe('Login', function(){


    });
}
```

- the first test case we are going to write is one that makes sure that setting the error state on the Login component actually works correctly

```
if(Meteor.isClient){
    describe('Login', function(){
        it('should show error messages', function(){


        });
    });
}
```

- the fist thing we need to do is to define an error message that we going to use through out our test case

```
if(Meteor.isClient){
    describe('Login', function(){
        it('should show error messages', function(){
            const error = 'this is not working';
        });
    });
}
```

- next is render Login, we gotta make sure it sets it state and make sure its working, so we need an instance of the Login component

```
if(Meteor.isClient){
    describe('Login', function(){
        it('should show error messages', function(){
            const error = 'this is not working';
            const wrapper = mount();
        });
```

```
    });
}
```

- lets now import the Login component above

```
import { Meteor } from 'meteor/meteor';
import React from 'react';
import expect from 'expect';
import { mount } from 'enzyme';

import { Login } from './Login'

if(Meteor.isClient){
    describe('Login', function(){
        it('should show error messages', function(){
            const error = 'this is not working';
            const wrapper = mount();
        });
    });
}
```

- inside of mount we can make an instance of Login component, Login takes a prop, which is loginWithPassword and we are going to set that to an empty function

```
if(Meteor.isClient){
    describe('Login', function(){
        it('should show error messages', function(){
            const error = 'this is not working';
            const wrapper = mount(<Login loginWithPassword={()=>{}}/>);
        });
    });
}
```

- now we need to set a state to that component. setState take one argument, an object

```
if(Meteor.isClient){

    describe('Login', function(){

        it('should show error messages', function(){

            const error = 'this is not working';

            const wrapper = mount(<Login loginWithPassword={()=>{}}/>);


            wrapper.setState({});

        });

    });

}
```

- and we want to set the error state so its going to be error: error, because we are setting it it the const error up above, but we are going to use es6 shorthand for this one

```
if(Meteor.isClient){

    describe('Login', function(){

        it('should show error messages', function(){

            const error = 'this is not working';

            const wrapper = mount(<Login loginWithPassword={()=>{}}/>);


            wrapper.setState({ error });

        });

    });

}
```

- now we are going to write our assertion
- first we are going to select wrapper p tags, then get the text value
- then expect() to be equal to error, the variable above

```
import { Meteor } from 'meteor/meteor';

import React from 'react';

import expect from 'expect';

import { mount } from 'enzyme';
```

```
import { Login } from './Login'


if(Meteor.isClient){
    describe('Login', function(){
        it('should show error messages', function(){
            const error = 'this is not working';
            const wrapper = mount(<Login loginWithPassword={()=>{}}/>);


            wrapper.setState({ error });
            const p = wrapper.find('p').text();
            expect(p).toBe(error);
        });
    });
}
```

- next is clear this state, and make sure there is no paragraph tag, we are going to setState one more time

```
if(Meteor.isClient){
    describe('Login', function(){
        it('should show error messages', function(){
            const error = 'this is not working';
            const wrapper = mount(<Login loginWithPassword={()=>{}}/>);


            wrapper.setState({ error });
            const p = wrapper.find('p').text();
            expect(p).toBe(error);

            wrapper.setState({ error: ''});
        });
    });
}
```

- now we are goin to assert, and we want to check there are no p tags

```
import { Meteor } from 'meteor/meteor';
import React from 'react';
import expect from 'expect';
import { mount } from 'enzyme';

import { Login } from './Login'

if(Meteor.isClient){
    describe('Login', function(){
        it('should show error messages', function(){
            const error = 'this is not working';
            const wrapper = mount(<Login loginWithPassword={()=>{}}/>);

            wrapper.setState({ error });
            const p = wrapper.find('p').text();
            expect(p).toBe(error);

            wrapper.setState({ error: ''});
             expect( wrapper.find('p').length).toBe(0);
        });
    });
}
```

-