130-Testing Setup video

- go to Signup.js and import

```
import React from 'react';

import { Link } from 'react-router';

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

import { createContainer } from 'meteor/react-meteor-data';
```

- **now lets** swith from default export to named export

```
export class Signup extends React.Component{

    constructor(props){

        super(props);

        this.state = {

            error: ' '

        };


    }
```

- and down below we are going to do the default export createContainer, and we pass in the two arguments we always use for createContainer, which is a function, and the componenet we are trying to containerized which is Signup

```
import React from 'react';

import { Link } from 'react-router';

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

import { createContainer } from 'meteor/react-meteor-data';


export class Signup 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();

        if(password.length < 9){
            return this.setState({error: 'Password must be more than 8 characters long.'})
        }

        Accounts.createUser({email, password}, (err)=>{
           if(err){
            this.setState({error: err.reason});
           }else{
            this.setState({error: ''});
           }
        });

    }
    render(){
        return (
            <div className = "boxed-view">
                <div className = "boxed-view__box">
                    <h1>Join</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="Passw
ord"/>

                        <button className="button">Create Account</button>

                    </form>


                    <Link to="/">Already have an account?</Link>


                </div>

            </div>

        );

    }

}


export default createContainer(()=>{


},Signup);
```

-
- **and inside the only prop we are going to pass is the createUser**

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

    return {

        createUser : Accounts.createUser

    }

},Signup);
```

- now lets setup a propType for Signup up above our default export

```
Signup.propTypes = {

    createUser : React.PropTypes.func.isRequired

};


export default createContainer(()=>{

    return {
        createUser : Accounts.createUser

    }

},Signup);
```

- now change the Accounts.createUser to this.props

```
import React from 'react';

import { Link } from 'react-router';

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

import { createContainer } from 'meteor/react-meteor-data';


export class Signup 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();


        if(password.length < 9){

            return this.setState({error: 'Password must be more than 8 characters long.'})

        }


        this.props.createUser({email, password}, (err)=>{

           if(err){

            this.setState({error: err.reason});

           }else{
            this.setState({error: ''});
           }
        });

    }
    render(){
        return (
            <div className = "boxed-view">
                <div className = "boxed-view__box">
                    <h1>Join</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">Create Account</button>
                    </form>

                    <Link to="/">Already have an account?</Link>

                </div>
            </div>
        );
    }
}

Signup.propTypes = {
    createUser : React.PropTypes.func.isRequired
};
```

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

    return {

        createUser : Accounts.createUser

    }

},Signup);
```

- we are done setting up this component, check the browser to see is everything is still working
- lets run our app from test to development mode

```
joses-MacBook-Pro:notes mendoza$ meteor
```

- **create an account, and make sure everthing is working, logout, on go the terminal and switch to test mode**

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

- **over in imports/ui create a file called Signup.test.js**
- **since Login.test.js is similar to our new file, lets copy everything from Login.test.js and paste it onto Login.test.js**

```
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);
        });

        it('should call loginWithPassword with the form data', function(){
            const email = 'jose@test.com';
            const password = 'password123';
            const spy = expect.createSpy();
            const wrapper = mount(<Login loginWithPassword={spy}/>);

            wrapper.ref('email').node.value = email;
            wrapper.ref('password').node.value = password;
            wrapper.find('form').simulate('submit');

            expect(spy.calls[0].arguments[0]).toEqual({ email });
            expect(spy.calls[0].arguments[1]).toBe(password);



        });

        it('should set loginWithPassword callback error', function(){
            const spy = expect.createSpy();
            const wrapper = mount(<Login loginWithPassword={spy}/>);

            wrapper.find('form').simulate('submit');

            spy.calls[0].arguments[2]({});
            expect(wrapper.state('error').length).toNotBe(0);

            spy.calls[0].arguments[2]();
            expect(wrapper.state('error').length).toBe(0);
```

```
        })

    });

}
```

- and now lets start changing some things
- first thing is the import, also on describe switch it Signup, next we want to render our Signup component and the props is going to be createUser, next comment out the two test cases for now

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

import React from 'react';

import expect from 'expect';

import { mount } from 'enzyme';


import { Signup } from './Signup'


if(Meteor.isClient){

    describe('Signup', function(){

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

            const error = 'this is not working';

            const wrapper = mount(<Signup createUser={()=>{}}/>);


            wrapper.setState({ error });

            const p = wrapper.find('p').text();

            expect(p).toBe(error);


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

            expect( wrapper.find('p').length).toBe(0);

        });


        // it('should call loginWithPassword with the form data', function(){

        //     const email = 'jose@test.com';

        //     const password = 'password123';

        //     const spy = expect.createSpy();

        //     const wrapper = mount(<Login loginWithPassword={spy}/>);
        //     wrapper.ref('email').node.value = email;

        //     wrapper.ref('password').node.value = password;

        //     wrapper.find('form').simulate('submit');


        //     expect(spy.calls[0].arguments[0]).toEqual({ email });

        //     expect(spy.calls[0].arguments[1]).toBe(password);




        // });


        // it('should set loginWithPassword callback error', function(){

        //     const spy = expect.createSpy();

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


        //     wrapper.find('form').simulate('submit');


        //     spy.calls[0].arguments[2]({});

        //     expect(wrapper.state('error').length).toNotBe(0);


        //     spy.calls[0].arguments[2]();

        //     expect(wrapper.state('error').length).toBe(0);

        // })

    });

}
```

- and first case is passing
- now lets uncomment the second test and start changing a few things, first thing the test should call createUser, and we also want an instance Signup, with createUser as the prop, and in our assertion we are going to be checking for email and password in our first argument

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

import React from 'react';

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


import { Signup } from './Signup'


if(Meteor.isClient){

    describe('Signup', function(){

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

            const error = 'this is not working';

            const wrapper = mount(<Signup createUser={()=>{}}/>);


            wrapper.setState({ error });

            const p = wrapper.find('p').text();

            expect(p).toBe(error);


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

            expect( wrapper.find('p').length).toBe(0);

        });


        it('should call createUser with the form data', function(){

            const email = 'jose@test.com';

            const password = 'password123';

            const spy = expect.createSpy();

            const wrapper = mount(<Signup createUser={spy}/>);


            wrapper.ref('email').node.value = email;

            wrapper.ref('password').node.value = password;

            wrapper.find('form').simulate('submit');


            expect(spy.calls[0].arguments[0]).toEqual({ email, password });


        });
```

- now we are going to do a test cases for the password length, so lets copy the above test case and paste right below it, and then we'll start changing it

```
        it('should call createUser with the form data', function(){

            const email = 'jose@test.com';

            const password = 'password123';

            const spy = expect.createSpy();

            const wrapper = mount(<Signup createUser={spy}/>);


            wrapper.ref('email').node.value = email;

            wrapper.ref('password').node.value = password;

            wrapper.find('form').simulate('submit');


            expect(spy.calls[0].arguments[0]).toEqual({ email, password });


        });

        it('should call createUser with the form data', function(){

            const email = 'jose@test.com';

            const password = 'password123';

            const spy = expect.createSpy();

            const wrapper = mount(<Signup createUser={spy}/>);


            wrapper.ref('email').node.value = email;

            wrapper.ref('password').node.value = password;

            wrapper.find('form').simulate('submit');


            expect(spy.calls[0].arguments[0]).toEqual({ email, password });


        });
```

now this test case should be to should set error if short password , then lets change our const password to a value with less then 9 characters, then we will change assertion as well

```
        it('should set error if short password', function(){
            const email = 'jose@test.com';

            const password = '123                ';

            const spy = expect.createSpy();

            const wrapper = mount(<Signup createUser={spy}/>);


            wrapper.ref('email').node.value = email;

            wrapper.ref('password').node.value = password;

            wrapper.find('form').simulate('submit');


            expect(wrapper.state('error').length).toBeGreaterThan(0);


        });
```

- now lets uncomment that last test case and start making some changes-first it should set createUser callback error, next create a const password with a valid length, next our component we are mounting is Signup with createUser as the prop, and dont'forget to set the password
-

```
        it('should set createUser callback error', function(){

            const password = 'password123!'

            const spy = expect.createSpy();

            const wrapper = mount(<Signup createUser={spy}/>);


            wrapper.ref('password').node.value = password;

            wrapper.find('form').simulate('submit');


            spy.calls[0].arguments[2]({});

            expect(wrapper.state('error').length).toNotBe(0);


            spy.calls[0].arguments[2]();

            expect(wrapper.state('error').length).toBe(0);

        })

    });

}
```

- now when we use spy.calls we can't just use an empty object, we need to pass in the reason variable, so lets create up above first

```
    it('should set createUser callback error', function(){
        const password = 'password123!'
        const reason = 'this is why is failed';
        const spy = expect.createSpy();
        const wrapper = mount(<Signup createUser={spy}/>);

        wrapper.ref('password').node.value = password;
        wrapper.find('form').simulate('submit');

        spy.calls[0].arguments[2]({});
        expect(wrapper.state('error').length).toNotBe(0);

        spy.calls[0].arguments[2]();
        expect(wrapper.state('error').length).toBe(0);
    })
```

- then we are expecting the second argument and this time we are not chekcing the length but just the value

```
  it('should set createUser callback error', function(){
        const password = 'password123!'
        const reason = 'this is why is failed';
        const spy = expect.createSpy();
        const wrapper = mount(<Signup createUser={spy}/>);

        wrapper.ref('password').node.value = password;
        wrapper.find('form').simulate('submit');

        spy.calls[0].arguments[1]({ reason });
        expect(wrapper.state('error')).toBe(reason);
```

```
            spy.calls[0].arguments[1]();
            expect(wrapper.state('error').length).toBe(0);
        })
    });
}
```

-