146-Loading Newly created Notes [video]

- we are going to start by going to NoteList.js and we want our newly creted notes to be at the top, and that is going to happen as an argument in Notes.find()

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


    const selectedNoteId = Session.get('selectedNoteId');


    Meteor.subscribe('notes');


    return {
        notes : Notes.find().fetch().map((note)=>{
            return{
                ...note,
                selected: note._id === selectedNoteId
            }
        })
    };


}, NoteList);
```

- so we are going to pass and empty object in our first argument in find() so it can go through the whole database, we then provide the second argument another object but with sort as our prop, and give this the value of -1 which in this case mean decending

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


    const selectedNoteId = Session.get('selectedNoteId');


    Meteor.subscribe('notes');


    return {
        notes : Notes.find({},{
            sort:{

                updatedAt: -1

            }

        }).fetch().map((note)=>{

            return{

                ...note,

                selected: note._id === selectedNoteId

            }

        })

    };


}, NoteList);
```

- there is a but with this solution so lets go to NoteListHeader.js to fix it, we are going to provide a call back function to props.meteorCall

```
import React from 'react';

import { Meteor } from 'meteor/meteor';

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

import PropTypes from 'prop-types';


export const NoteListHeader = (props)=>{

    return (

        <div>

            <button onClick = {()=>{

                props.meteorCall('notes.insert');

                }}>Create Note</button>

        </div>

    );

};
```

- we are going to pass two arguments, an err, or a res

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

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

import PropTypes from 'prop-types';


export const NoteListHeader = (props)=>{

    return (

        <div>

            <button onClick = {()=>{

                props.meteorCall('notes.insert',(err, res)=>{


                });

                }}>Create Note</button>

        </div>

    );

};
```

- and we are going to be needing Session so lets import it

```
import React from 'react';

import { Meteor } from 'meteor/meteor';

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

import { Session } from 'meteor/session';

import PropTypes from 'prop-types';
```

- and Session in a required propType object

```
NoteListHeader.propTypes = {

    meteorCall : PropTypes.func.isRequired,

    Session: PropTypes.object.isRequired

};


export default createContainer(()=>{

    return {
        meteorCall : Meteor.call,

        Session

    };

},NoteListHeader);
```

- now lets set this Session, and this case we don't need the 'this' keyword because this is a stateless component

```
import React from 'react';

import { Meteor } from 'meteor/meteor';

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

import { Session } from 'meteor/session';

import PropTypes from 'prop-types';


export const NoteListHeader = (props)=>{

    return (

        <div>

            <button onClick = {()=>{

                props.meteorCall('notes.insert',(err, res)=>{

                    if(res){

                        props.Session.set('selectedNoteId', res)

                    }

                });

                }}>Create Note</button>

        </div>

    );

};
```

- we now want to create a test case the Session.set gets called correctly so NoteListHeader.test.js
- we are going to be refactoring some of this code since we have new techniques, first lets import notes from fixtures up above

```
import React from 'react';

import expect from 'expect';

import  { mount } from 'enzyme';
import { Meteor } from 'meteor/meteor';

import { notes } from '../fixtures/fixtures'
```

- **lets now create some consts in the describe block**

```
import React from 'react';

import expect from 'expect';

import  { mount } from 'enzyme';

import { Meteor } from 'meteor/meteor';

import { notes } from '../fixtures/fixtures'


import { NoteListHeader } from './NoteListHeader';


if(Meteor.isClient){

    describe('NoteListHeader', function(){


        let meteorCall;

        let Session;


        it('should call meteorCall on click', function(){

            const spy = expect.createSpy();

            const wrapper = mount(<NoteListHeader meteorCall={spy}/>);


            wrapper.find('button').simulate('click');

            expect(spy).toHaveBeenCalledWith('notes.insert');

        });

    });

}
```

- and we can define the in the beforeEach method - meteorCall will be the spy and Session will a prop of set and that will be a spy as well

```
import React from 'react';

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

import { Meteor } from 'meteor/meteor';

import { notes } from '../fixtures/fixtures'


import { NoteListHeader } from './NoteListHeader';


if(Meteor.isClient){

    describe('NoteListHeader', function(){


        let meteorCall;

        let Session;


        beforeEach(function(){

            meteorCall = expect.createSpy();

            Session = {

                set: expect.createSpy()

            }

        });

```

- **lets not modify our test case**

```
import React from 'react';

import expect from 'expect';

import  { mount } from 'enzyme';

import { Meteor } from 'meteor/meteor';

import { notes } from '../fixtures/fixtures'


import { NoteListHeader } from './NoteListHeader';


if(Meteor.isClient){

    describe('NoteListHeader', function(){

        let meteorCall;

        let Session;


        beforeEach(function(){

            meteorCall = expect.createSpy();

            Session = {

                set: expect.createSpy()

            }

        });


        it('should call meteorCall on click', function(){


            const wrapper = mount(<NoteListHeader meteorCall={meteorCall} Session = {Session}

/>);


            wrapper.find('button').simulate('click');

            expect(spy).toHaveBeenCalledWith('notes.insert');

        });

    });

}
```

- we then are going to modify the expect() method

```
        it('should call meteorCall on click', function(){


            const wrapper = mount(<NoteListHeader meteorCall={meteorCall} Session = {Session}

/>);


            wrapper.find('button').simulate('click');

            expect(meteorCall.calls[0].arguments[0]).toBe('notes.insert');

        });
```

- now it should be passing all tests
- and now we are going to test the second argument

```
        it('should call meteorCall on click', function(){


            const wrapper = mount(<NoteListHeader meteorCall={meteorCall} Session = {Session}
/>);


            wrapper.find('button').simulate('click');

            meteorCall.calls[0].arguments[1](undefined, notes[0]._id);

            expect(meteorCall.calls[0].arguments[0]).toBe('notes.insert');

        });

    })
```

- now lets make the assertion

```
 it('should call meteorCall on click', function(){


            const wrapper = mount(<NoteListHeader meteorCall={meteorCall} Session = {Session}
/>);


            wrapper.find('button').simulate('click');

            meteorCall.calls[0].arguments[1](undefined, notes[0]._id);

            expect(meteorCall.calls[0].arguments[0]).toBe('notes.insert');

            expect(Session.set).toHaveBeenCalledWith('selectedNoteId', notes[0]._id);

        });
```

- the next test will be for a failed set Session

```
        it('should call meteorCall on click', function(){


            const wrapper = mount(<NoteListHeader meteorCall={meteorCall} Session = {Session}
/>);


            wrapper.find('button').simulate('click');

            meteorCall.calls[0].arguments[1](undefined, notes[0]._id);

            expect(meteorCall.calls[0].arguments[0]).toBe('notes.insert');
            expect(Session.set).toHaveBeenCalledWith('selectedNoteId', notes[0]._id);

        });


        it('should not set session for failed insert', function(){


                const wrapper = mount(<NoteListHeader meteorCall={meteorCall} Session = {Session}
    />);


                wrapper.find('button').simulate('click');

                meteorCall.calls[0].arguments[1]({}, undefined);

                expect(meteorCall.calls[0].arguments[0]).toBe('notes.insert');

                expect(Session.set).toNotHaveBeenCalled();

          });
```

- **lets now commit and push our code**

```
joses-MacBook-Pro:notes mendoza$ git add .

joses-MacBook-Pro:notes mendoza$ git commit -m "Select newly created note.Sort notes by updatedA
t"

joses-MacBook-Pro:notes mendoza$ git push
```

-