143-Removing Notes video

- there seems to be a bug when typing the title of a note, we are going to fix that
- go the Editor.js
- we are going to be using State, so that means we are going to be using a constructor

```
import React from 'react';

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

import { Session } from 'meteor/session';

import { Meteor } from 'meteor/meteor';

import { Notes } from '../api/notes';

import PropTypes from 'prop-types'


export class Editor extends React.Component{

    constructor (props){

        super(props)

        this.state = {

            title: ' ',

            body: ''

        }

    }
```

- **no**w inside of handleBodyChange lets create a const body and give it the value of the target.value, and we then set the state to it

```
import React from 'react';

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

import { Session } from 'meteor/session';

import { Meteor } from 'meteor/meteor';

import { Notes } from '../api/notes';

import PropTypes from 'prop-types'


export class Editor extends React.Component{
```

```
    constructor (props){
        super(props)
        this.state = {
            title: ' ',
            body: ''
        }
    }

    handleBodyChange(e){
        const body = e.target.value;
        this.setState({body});
        this.props.call('notes.update', this.props.note._id, {
            body: e.target.value
        });
    }
```

- and we will do the same thing when we use props.call

```
    handleBodyChange(e){
        const body = e.target.value;
        this.setState({body});
        this.props.call('notes.update', this.props.note._id, { body });
    }
```

- we are going to be doing the same thing with handleTitleChange

```
    handleBodyChange(e){
        const body = e.target.value;
        this.setState({body});
        this.props.call('notes.update', this.props.note._id, { body });
    }

    handleTitleChange(e){
```

```
        const title = e.target.value;
        this.setState({title});
        this.props.call('notes.update', this.props.note._id, {title})
    }
```

- now lets render all of this down below

```
    handleTitleChange(e){
        const title = e.target.value;
        this.setState({title});
        this.props.call('notes.update', this.props.note._id, {title})
    }

    render(){
        if(this.props.note){
            return (
              <div>
                  <input value={this.state.title} placeholder="Untitled" onChange={this.handleTi
tleChange.bind(this)}/>
                  <textarea value={this.state.body} placeholder="Your note here" onChange=
{this.handleBodyChange.bind(this)}></textarea>
                  <button></button>
              </div>
```

- now the cursor issue is working propertly and is not jumping to the end like it used, but we still have some issues, we are going to be using lifecycle method to fix this and it is componentDidUpdate

```
    handleBodyChange(e){
        const body = e.target.value;
        this.setState({body});
        this.props.call('notes.update', this.props.note._id, { body });
    }
```

```
handleTitleChange(e){
    const title = e.target.value;
    this.setState({title});
    this.props.call('notes.update', this.props.note._id, {title})
}

componentDidUpdate(prevProps, prevState){

}
```

- **in this method** we are going to check wheather a note exists, if it does then we will give it an id if it doesnt then it will be undefined, samething with title

```
handleBodyChange(e){
    const body = e.target.value;
    this.setState({body});
    this.props.call('notes.update', this.props.note._id, { body });
}

handleTitleChange(e){
    const title = e.target.value;
    this.setState({title});
    this.props.call('notes.update', this.props.note._id, {title})
}

componentDidUpdate(prevProps, prevState){
    const currentNoteId = this.props.note ? this.props.note._id : undefined;
    const prevNoteId = prevProps.note ? prevProps.note_id : undefined;
}
```

- we now are ready to set the state, but before that we need to check if there is a currentNote and also if the currentNoteId is not the previous noteId

```
componentDidUpdate(prevProps, prevState){
```

```
        const currentNoteId = this.props.note ? this.props.note._id : undefined;
        const prevNoteId = prevProps.note ? prevProps.note_id : undefined;

        if(currentNoteId && currentNoteId !== prevNoteId){
            this.setState({
                title: this.props.note.title,
                body: this.props.note.body
            });
        }
    }
```

- now in your browser the cursor issue should be resolve as well as the text gets preserved even if you jump from one note to another
- next we want to wire up the delete button

```
    render(){
        if(this.props.note){
            return (
              <div>
                  <input value={this.state.title} placeholder="Untitled" onChange={this.handleTi
tleChange.bind(this)}/>
                  <textarea value={this.state.body} placeholder="Your note here" onChange=
{this.handleBodyChange.bind(this)}></textarea>
                  <button>Delete note</button>
              </div>
```

- we are going to delete notes now
- first lets give an onClick event and call a method we havent yet created

```
    render(){
        if(this.props.note){
            return (
              <div>
                  <input value={this.state.title} placeholder="Untitled" onChange={this.handleTi
tleChange.bind(this)}/>
```

```
                <textarea value={this.state.body} placeholder="Your note here" onChange=
{this.handleBodyChange.bind(this)}></textarea>
                <button onClick={this.handleRemoval.bind(this)}>Delete note</button>
              </div>
            );
```

- lets then create this function above

```
    handleRemoval(){

    }

    componentDidUpdate(prevProps, prevState){
        const currentNoteId = this.props.note ? this.props.note._id : undefined;
        const prevNoteId = prevProps.note ? prevProps.note_id : undefined;

        if(currentNoteId && currentNoteId !== prevNoteId){
            this.setState({
                title: this.props.note.title,
                body: this.props.note.body
            });
        }
    }

    render(){
        if(this.props.note){
            return (
              <div>
                <input value={this.state.title} placeholder="Untitled" onChange={this.handleTi
tleChange.bind(this)}/>
                <textarea value={this.state.body} placeholder="Your note here" onChange=
{this.handleBodyChange.bind(this)}></textarea>
                <button onClick={this.handleRemoval.bind(this)}>Delete note</button>
```

```
            </div>
          );
```

- lets call notes.remove and we know it requires one argument

```
  handleRemoval(){
      this.props.call('notes.remove', this.props.note._id);
    }
```

- this works but the url remains the same, so we the messege note not found, we don't want that, we want the message that says to create a new note
- we are going to achieve this by using browserHistory like we did before, so lets import it first

```
import React from 'react';
import { createContainer } from 'meteor/react-meteor-data';
import { Session } from 'meteor/session';
import { Meteor } from 'meteor/meteor';
import { browserHistory } from 'react-router';
import { Notes } from '../api/notes';
import PropTypes from 'prop-types'
```

- now lets setup down below for testing purposes and also in propTypes

```
Editor.propTypes = {
    note : PropTypes.object,
    selectedNoteId : PropTypes.string,
    browserHistory: PropTypes.object.isRequired
}

export default createContainer(()=>{
    const selectedNoteId = Session.get('selectedNoteId');

    return {
        selectedNoteId,
```

```
        note: Notes.findOne(selectedNoteId),
        call: Meteor.call,
        browserHistory
    };
}, Editor)
```

- and now lets set it up in our handleRemove, so what we want is that when we delete a note, we we want our browserHistory to go to the homepage dashboard

```
    handleRemoval(){
        this.props.call('notes.remove', this.props.note._id);
        this.props.browserHistory.push('/dashboard');
    }
```

- and in the browser you should see the message Pick or create a note to get started
- we are now going to commit and push our code
- this time we are going to use a shortcut to not use git add .

```
joses-MacBook-Pro:notes mendoza$ git commit -a -m "Setup state for title/body. Add handleRemove
 method"
joses-MacBook-Pro:notes mendoza$ git push
```