25-short-lnk-app *video*

Hiding Links With Method Methods

- go to links.js
- we are going to add a visible property on Links.insert

```
Meteor.methods({
    'links.insert'(url){
        if(!this.userId){
            throw new Meteor.Error('not-authorized')
        }
        new SimpleSchema({
            url:{
            type: String,
            label:'Your link',
            regEx: SimpleSchema.RegEx.Url
            }
        }).validate({ url });

        Links.insert({
            _id: shortid.generate(),
            url,
            userId: this.userId,
            visible: true
        });
    }
});
```

- now lets go to LinksListItem.js and we are going to add a new propType
- and now lets show it on the browser by putting in <p> tags

```
import React from 'react';
import Clipboard from 'clipboard';

export default class LinksListItem extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            justCopied : false
        }
    }

    componentDidMount(){
        this.clipboard = new Clipboard(this.refs.copy);
        this.clipboard.on('success', ()=>{
            this.setState({justCopied: true})
            setTimeout(()=>{
                this.setState({justCopied: false});
            },1000);
        }).on('error', ()=>{
            alert('please copy the link manually');
        });
    }
    componentWillUnmount(){
        this.clipboard.destroy();
    }
    render(){
        return(
            <div>
                <p>{this.props.url}</p>
                <p>{this.props.shortUrl}</p>
                <p>{this.props.visible.toString()}</p>
                <button ref="copy" data-clipboard-text={this.props.shortUrl}>
                {this.state.justCopied ? 'Copied' : 'Copy'}
                </button>
            </div>
        )
    }
}

LinksListItem.propTypes = {
    _id: React.PropTypes.string.isRequired,
    url: React.PropTypes.string.isRequired,

    userId: React.PropTypes.string.isRequired,

    visible: React.PropTypes.bool.isRequired,

    shortUrl: React.PropTypes.string.isRequired,
```

- we now want to make a button that has that functionality
- we then want to add our onClick handler
- we all also are going to be using Meteor.call() so we need to import Meteor in this page
- the .call() method will take two arguments - 1) links.setVisibility, which doesnt exist yet 2nd) this.props._id, and 3rd) !this.props.visible

```
import React from 'react';

import Clipboard from 'clipboard';

import { Meteor } from 'meteor/meteor';


export default class LinksListItem extends React.Component{


    constructor(props){

        super(props);

        this.state = {

            justCopied : false

        }

    }


    componentDidMount(){

        this.clipboard = new Clipboard(this.refs.copy);

        this.clipboard.on('success', ()=>{

            this.setState({justCopied: true})

            setTimeout(()=>{

                this.setState({justCopied: false});

            },1000);

        }).on('error', ()=>{

            alert('please copy the link manually');

        });

    }

    componentWillUnmount(){

        this.clipboard.destroy();

    }
    render(){
        return(
            <div>
                <p>{this.props.url}</p>
                <p>{this.props.shortUrl}</p>
                <p>{this.props.visible.toString()}</p>
                <button ref="copy" data-clipboard-text={this.props.shortUrl}>
                {this.state.justCopied ? 'Copied' : 'Copy'}
                </button>
                <button onClick={()=>{
                    Meteor.call('links.setVisibiltiy', this.props._id, !this.props.visible);
                }}>{this.props.visible ? 'Hide' : 'Unhide'}</button>
            </div>
        )
    }
}

LinksListItem.propTypes = {
    _id: React.PropTypes.string.isRequired,
    url: React.PropTypes.string.isRequired,
    userId: React.PropTypes.string.isRequired,
    visible: React.PropTypes.bool.isRequired,
    shortUrl: React.PropTypes.string.isRequired,

}
```

- now let create the links.setVisiblity method over in links.js
- over in links.js lets create the links.setVisibility function
- lets include the two argument _id and visible
- first lets check if the user is logged in, if not throw a meteor error
- we then are going to create a schema to validate the _id and visible and we are going to validate them
- we then are going to use Links.update where _id and this.userId match the doc
- and then we set the visible property to the visible argument

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';
import shortid from 'shortid';
```

```
export const Links = new Mongo.Collection('links');


if (Meteor.isServer){
    Meteor.publish('links', function(){
        return Links.find({userId : this.userId});
    })
}
Meteor.methods({
    'links.insert'(url){
        if(!this.userId){
            throw new Meteor.Error('not-authorized')
        }
        new SimpleSchema({
            url:{
            type: String,
            label:'Your link',
            regEx: SimpleSchema.RegEx.Url
            }
        }).validate({ url });


        Links.insert({
            _id: shortid.generate(),
            url,
            userId: this.userId,
            visible: true
        });
    },
    'links.setVisibility'(_id, visible){
        //check if user is logged in. Throw an error if not.
        if(!this.userId){
            throw new Meteor.Error('not-authorized');
        }
        new SimpleSchema({
            _id: {
                type: String,
                min: 1
            },
            visible:{
                type: Boolean
            }
        }).validate({_id, visible});

        Links.update({
            _id,
            userId: this.userId
        },{
            $set:{ visible }
        })
    }
});
```

- now when hitting the hide button over in the browser the boolean value should turn to false, and the button should turn to unhide
-