32-short-lnk-app video

Setting up Modal Errors

we are now going to handle the error when the user types something that isn't a url
lets go to AddLink.js
lets create another state property - by default it will be empty string

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

export default class AddLink extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            url: '',
            isOpen: false,
            error: ''
        }
    }
```

- first in the onSubmit method we are going to remove the if statement, not the code inside but just the if statement

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

export default class AddLink extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            url: '',
            isOpen: false,
            error: ''
        }
    }
```

```
    onSubmit(e){
        const { url } = this.state
        e.preventDefault();
        Meteor.call('links.insert', url, (err, res)=>{
            if(!err){
                this.setState({url:'', isOpen: false});
            }
        });
    }
```

- **lets** add an else clause
- and setState to error to err.reason
- and we are also going to clean it up if there is no error in the if statement
- and are also going to clear up the error when someone cancels
- and finally we are going to render the err.reason by using a paragraph
- lets also change the Add Link to an h1

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

export default class AddLink extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            url: '',
            isOpen: false,
            error: ''
        }
    }
    onSubmit(e){
        const { url } = this.state
        e.preventDefault();
        Meteor.call('links.insert', url, (err, res)=>{
            if(!err){
                this.setState({url:'', isOpen: false, error: ''});
            }else{
                this.setState({error: err.reason});
            }
```

```
        });
    }
    onChange(e){
        this.setState({
           url: e.target.value
        })
    }

    render(){
        return(
            <div>
                <button onClick={()=> this.setState({isOpen: true})}>+ Add Link</button>
                <Modal isOpen={this.state.isOpen}  contentLabel="Add link">
                    <h1>Add Link</h1>
                    {this.state.error ? <p>{this.state.error}</p> : undefined }
                    <form onSubmit={this.onSubmit.bind(this)}>
                        <input type="text" placeholder="URL" value={this.state.url} onChange={th
is.onChange.bind(this)}/>
                        <button>Add Link</button>
                    </form>
                    <button onClick={()=>{
                        this.setState({isOpen:false, url: '', error: ''});
                    }}>Cancel</button>
                </Modal>
            </div>
        );
    }
}
```

- to test it go to your browser and add a new link by not typing anything in the text field
- you should see a message above - *Your link must be a valid URL*
- we are going to do a technique where when the user hits the +add link the text field will automatically focus
- and we are going to be using a Modal property for this onAfterOpen
- and we also need to add the ref property to the input text field so we can reference it

```
    render(){
        return(
            <div>
```

```
                <button onClick={()=> this.setState({isOpen: true})}>+ Add Link</button>
                <Modal isOpen={this.state.isOpen}  contentLabel="Add link" onAfterOpen=
{()=>this.refs.url.focus()}>
                    <h1>Add Link</h1>
                    {this.state.error ? <p>{this.state.error}</p> : undefined }
                    <form onSubmit={this.onSubmit.bind(this)}>
                        <input ref="url" type="text" placeholder="URL" value={this.state.url} on
Change={this.onChange.bind(this)}/>
                        <button>Add Link</button>
                    </form>
                    <button onClick={()=>{
                        this.setState({isOpen:false, url: '', error: ''});
                    }}>Cancel</button>
                </Modal>
            </div>
        );
    }
}
```

- now another feature we want is when the modal is open when clicking outside of it we want the modal close
- and modal has a property for this onRequestClose
- lets define the method first

```
        this.setState({
           url: e.target.value
        })
    }

    handleModalClose(){
        this.setState({isOpen:false, url: '', error: ''});
    }

    render(){
        return(
            <div>
                <button onClick={()=> this.setState({isOpen: true})}>+ Add Link</button>
                <Modal isOpen={this.state.isOpen}  contentLabel="Add link" onAfterOpen=
```

```
{()=>this.refs.url.focus()}>
                    <h1>Add Link</h1>
```

- and we can use this function in a couple places where its needed

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

export default class AddLink extends React.Component{
    constructor(props){
        super(props);
        this.state = {
            url: '',
            isOpen: false,
            error: ''
        }
    }
    onSubmit(e){
        const { url } = this.state
        e.preventDefault();
        Meteor.call('links.insert', url, (err, res)=>{
            if(!err){
                this.handleModalClose();
            }else{
                this.setState({error: err.reason});
            }
        });
    }
    onChange(e){
        this.setState({
           url: e.target.value
        })
    }

    handleModalClose(){
```

```
        this.setState({isOpen:false, url: '', error: ''});
    }

    render(){
        return(
            <div>
                <button onClick={()=> this.setState({isOpen: true})}>+ Add Link</button>
                <Modal isOpen={this.state.isOpen}  contentLabel="Add link" onAfterOpen=
{()=>this.refs.url.focus()}
                onRequestClose={this.handleModalClose.bind(this)}>
                    <h1>Add Link</h1>
                    {this.state.error ? <p>{this.state.error}</p> : undefined }
                    <form onSubmit={this.onSubmit.bind(this)}>
                        <input ref="url" type="text" placeholder="URL" value={this.state.url} on
Change={this.onChange.bind(this)}/>
                        <button>Add Link</button>
                    </form>
                    <button onClick={this.handleModalClose.bind(this)}>Cancel</button>
                </Modal>
            </div>
        );
    }
}
```

- now we are going to add an ancher tag to our links
- lets go to LinksListItem
- we are going to be using the href attribute and the target attribute in our <a> tags

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

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();
    }
   renderStats(){
       const visitMessage = this.props.visitedCount === 1 ? 'visit' : 'visits';
       let visitedMessage = null;
       const momentNow = moment(this.props.lastVisitedAt);

       if(typeof this.props.lastVisitedAt === 'number'){
            visitedMessage = `(visited ${momentNow.fromNow()})`;
       }

       return <p>{this.props.visitedCount} {visitMessage} {visitedMessage}</p>
   }

    render(){
        return(
            <div>
                <p>{this.props.url}</p>
                <p>{this.props.shortUrl}</p>
                <p>{this.props.visible.toString()}</p>
                {this.renderStats()}
```

```
                <a href={this.props.shortUrl} target= "_blank">
                    Visit
                </a>
                <button ref="copy" data-clipboard-text={this.props.shortUrl}>
                {this.state.justCopied ? 'Copied' : 'Copy'}
                </button>
                <button onClick={()=>{
                    Meteor.call('links.setVisibility', 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,
    visitedCount: React.PropTypes.number.isRequired,
    lastVisitedAt: React.PropTypes.number
}
```

-
-
-
- <html
-
-