**Fixing a Small Bug**

The <dialog> element can also be closed by pressing the ESC key on the keyboard. In that case, the dialog will disappear but the state passed to the open prop (i.e., the modalIsOpen state) will not be set to false.

Therefore, the modal can't be opened again (because modalIsOpen still is true - the UI basically now is not in sync with the state anymore).

To fix this issue, we must listen to the modal being closed by adding the built-in onClose prop to the <dialog>. The event is then "forwarded" to the App component by accepting a custom onClose prop on the Modal component.

The Modal component therefore should look like this:

import { useRef, useEffect } from 'react';

import { createPortal } from 'react-dom';

function Modal({ open, children, onClose }) {

const dialog = useRef();

useEffect(() => {

if (open) {

dialog.current.showModal();

} else {

dialog.current.close();

}

}, [open]);

return createPortal(

<dialog className="modal" ref={dialog} onClose={onClose}>

{children}

</dialog>,

document.getElementById('modal')

);

}

export default Modal;

In the App component, we can now set the handleStopRemovePlace function as a value for the onClose prop on the <Modal> component:

<Modal open={modalIsOpen} onClose={handleStopRemovePlace}>

<DeleteConfirmation

onCancel={handleStopRemovePlace}

onConfirm={handleRemovePlace}

/>

</Modal>