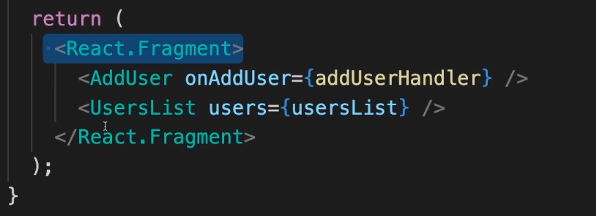
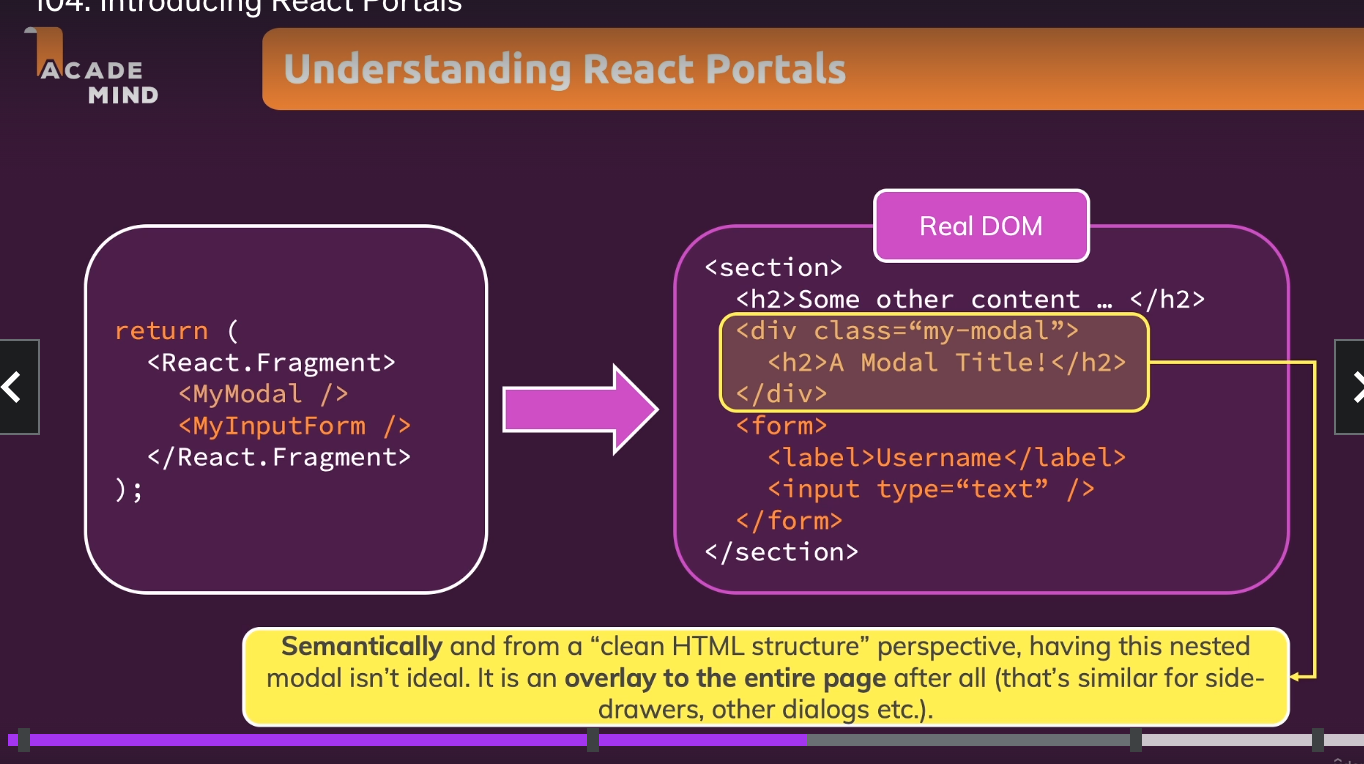
**6.Fragments, Portals & Refs**

1) JSX limitations :- adjacent root level elements will throw an error.

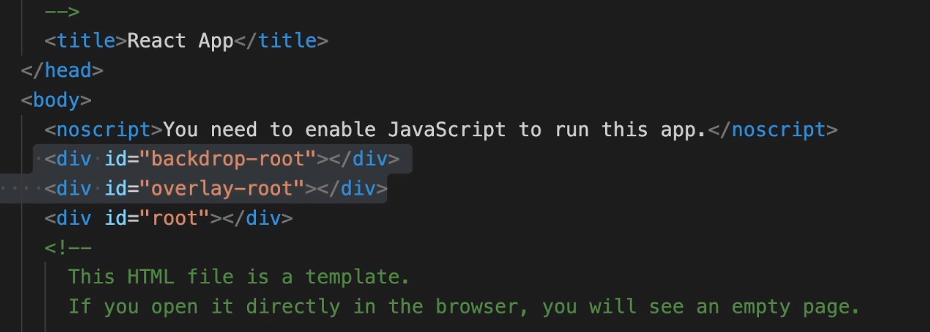
Solutions to the above problems is as follows:-

1. Use wrapping div. ( This ends with div soup ie. nested div inside each other also it could break styling and makes browser slower since we are rendering unnecessary content).
2. Use native javascript array . (we need a key in this case) (not commonly used solution)
3. Use a separate wrapper component as below that simply returns props.children and then enclose that wrapper component.
4. React has built in component called Fragmets and can we used as below. Or we can use <Fragment> directly.
5. 

**React portals :-**

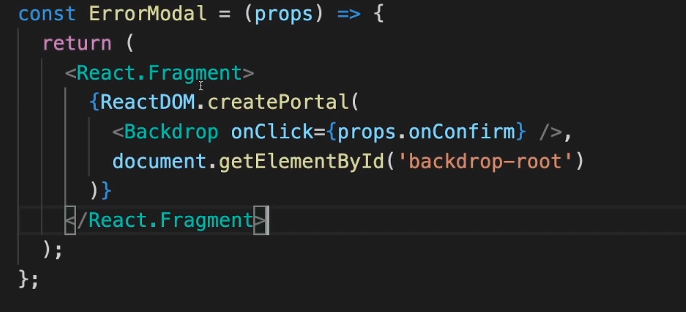


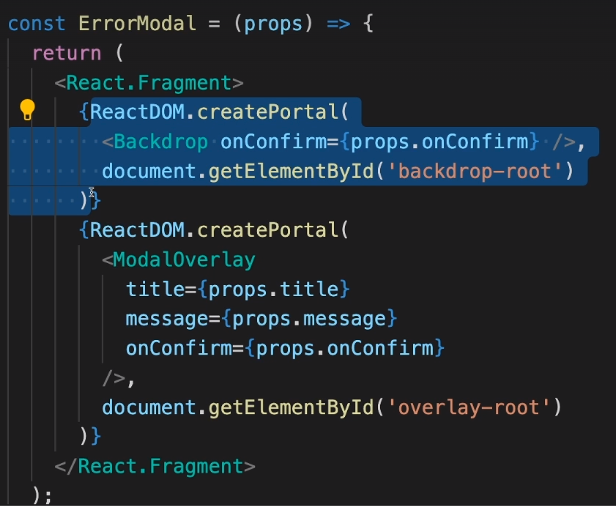
We can use portal to handle elements like a dailogue box etc. to be used in a common place somewhere else instead of using it inside the same component itself.

Steps for using portals :- For example in the root app we can have a separate div for backdrop root and modal root as follows:-

2) Create BackDrop and Modal components that render the JSX elements.

3) We need to import ReactDom for using portal.





**Working with refs**

With refs we can set up a connection between the HTML code and the Javascript code .

We use useRef() hook for the same.

For example if I want to connect an input HTML component with a ref we can do it with useRef hook.

Step 1:- use useRef() hook for example const nameInputRef = useRef();

Step 2:- Add ref = {nameInputRef} to that input element. Now refs are connected. Now the nameInputRef has the actual DOM node.

If we use refs , then that component would be an uncontrolled element since their internal state is not controlled by react.