React JS

It is opensource library for building User interfaces. It is not a frame work, it focuses on UI. It is capable of building full- fledged web applications.

It is created and maintained by Facebook (now Meta).

It is based on component -based architecture (components like header, footer). React is declarative i.e it means you just have to tell React, what type of UI has to be created, React will do it for you.

**File Structure of React**

package.json – It contains the list of dependencies and script to be used in the project.

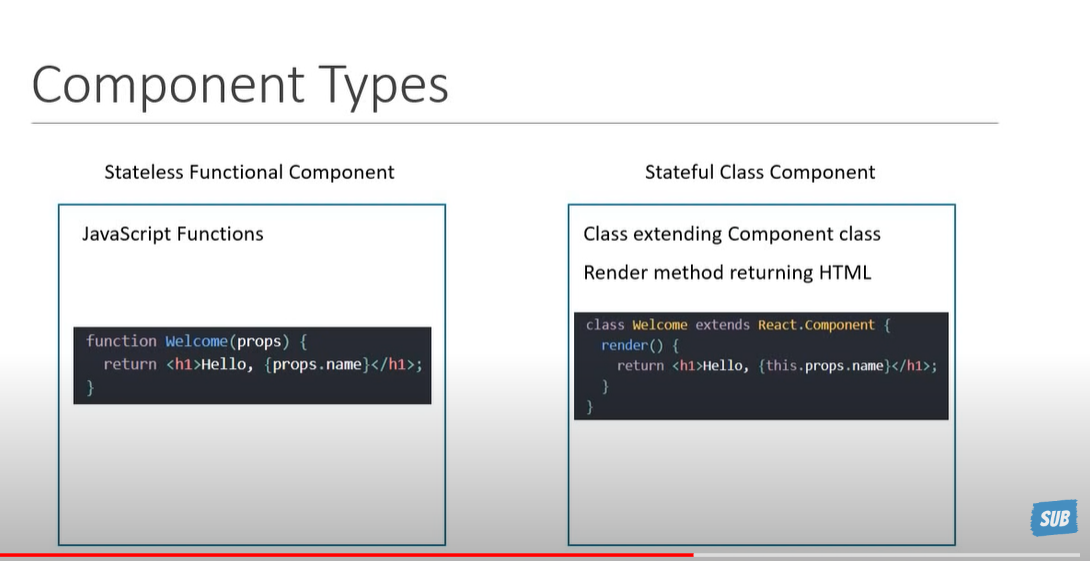
node-modules – All the dependencies are installed in this folder.

index.html (in public folder) – This is the only html file which is present in the entire react app, since it is used to develop single page applications, view changes in browser dynamically but index.html file remains the same. No changes need to be made in this file.

src - This folder contains the files in which most of the changes are done.

**Components**

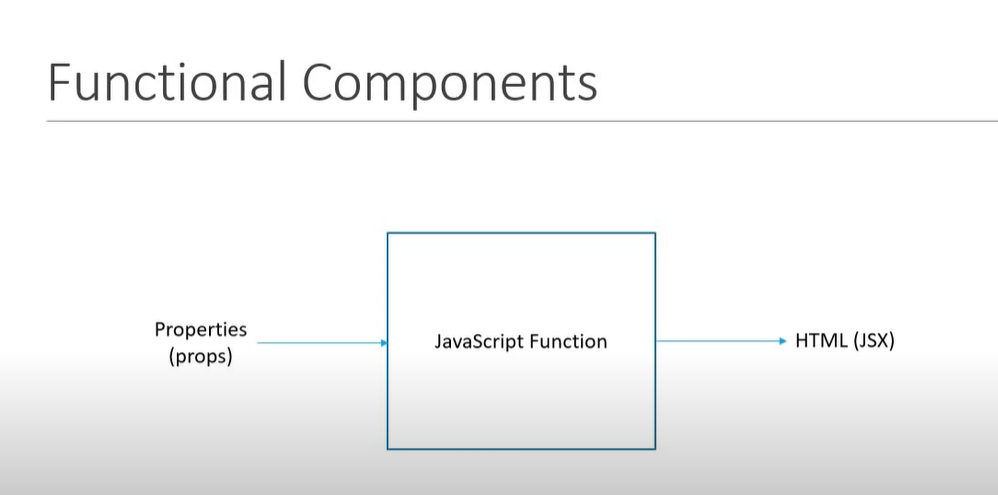
React contains many components like header, footer, side nav, main content. Root is the parent component. Components are re-usable and can be nested inside other components.

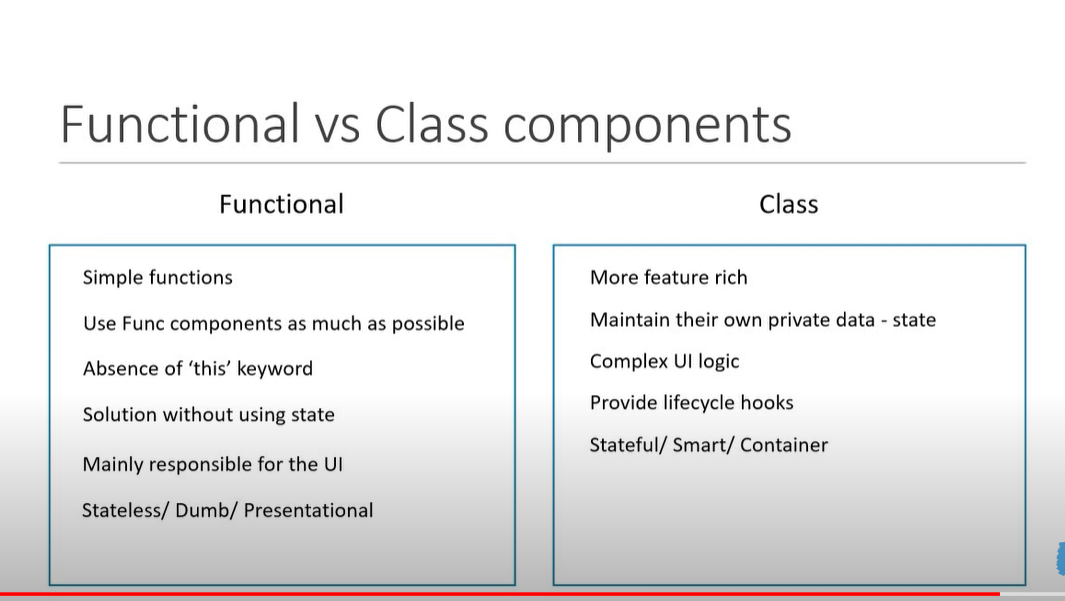


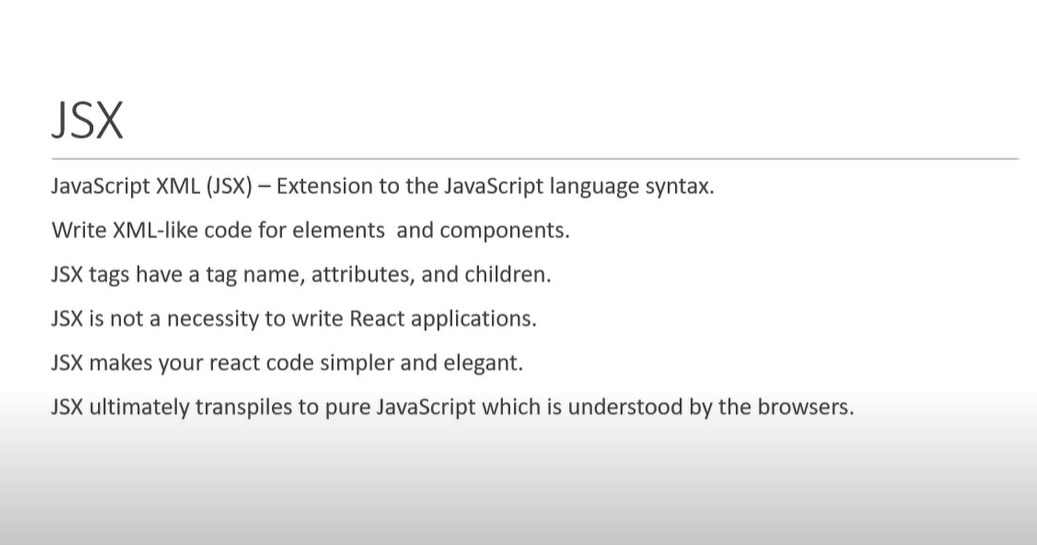
Type of Components:

1. Functional component

It takes in properties as an parameter to js function and return html which describes the UI.

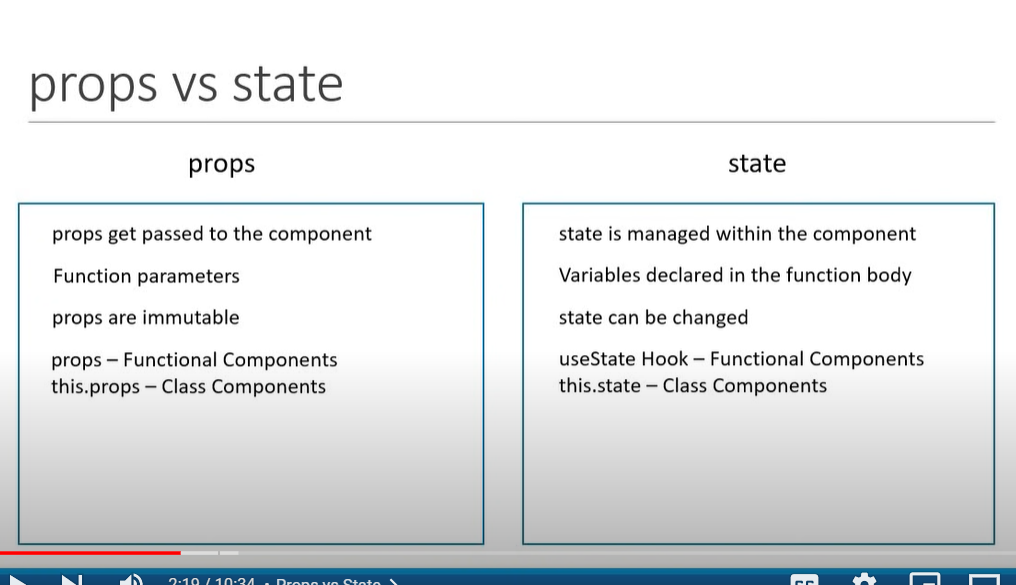






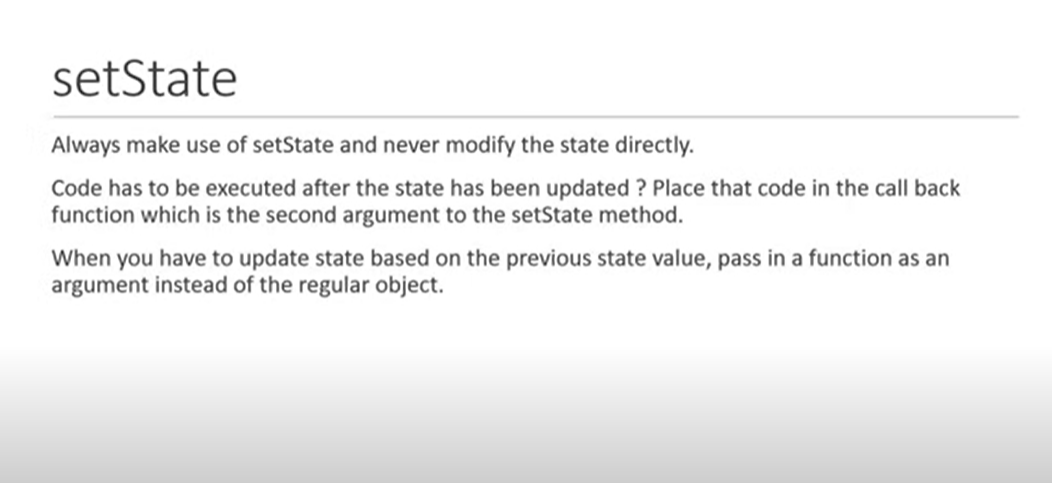
Props vs state

State can be changed within the component.

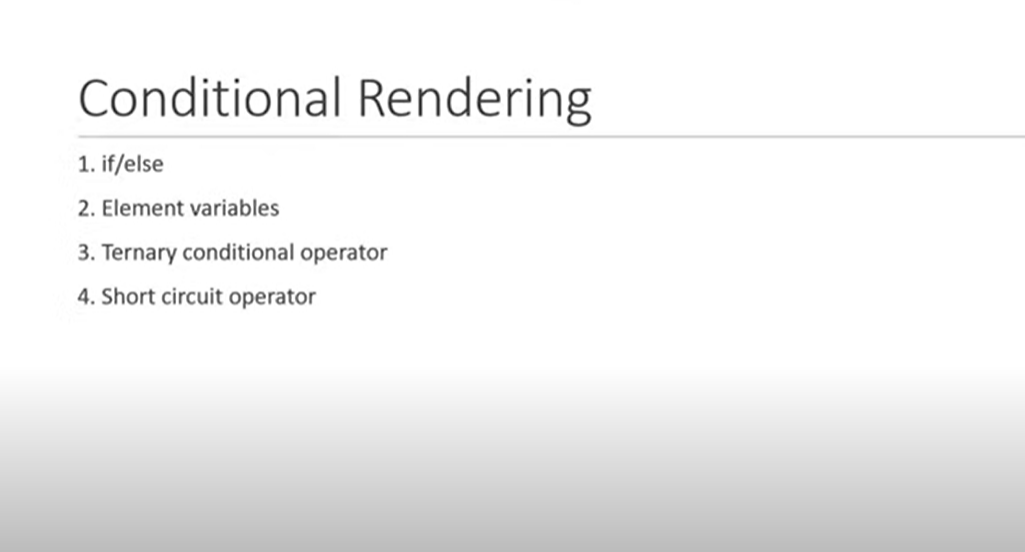


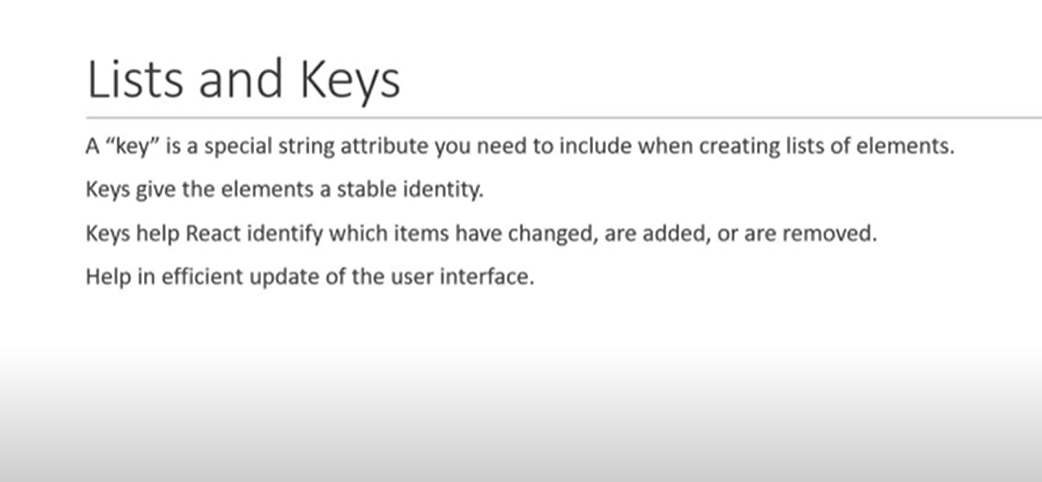
State should never be modified directly, setState should be used in order to modify the state.

In order to change the state, the previous state should be passed as an argument with callback function.

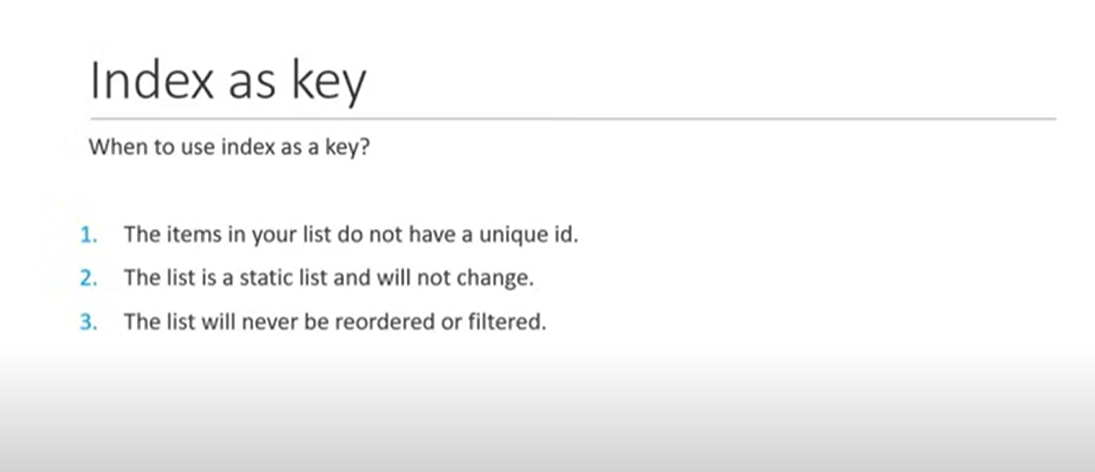


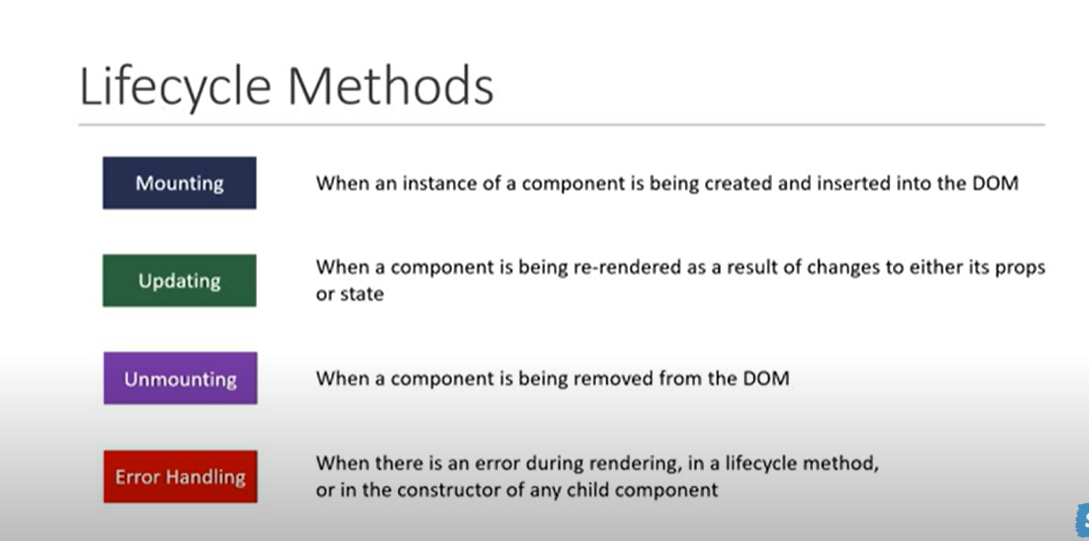
In Class Component , methods are accessed using this keyword.

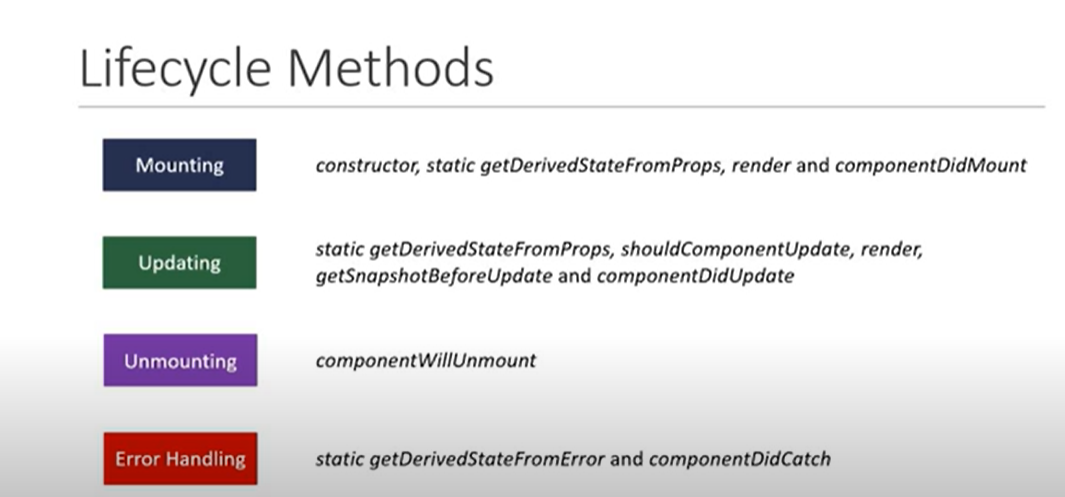


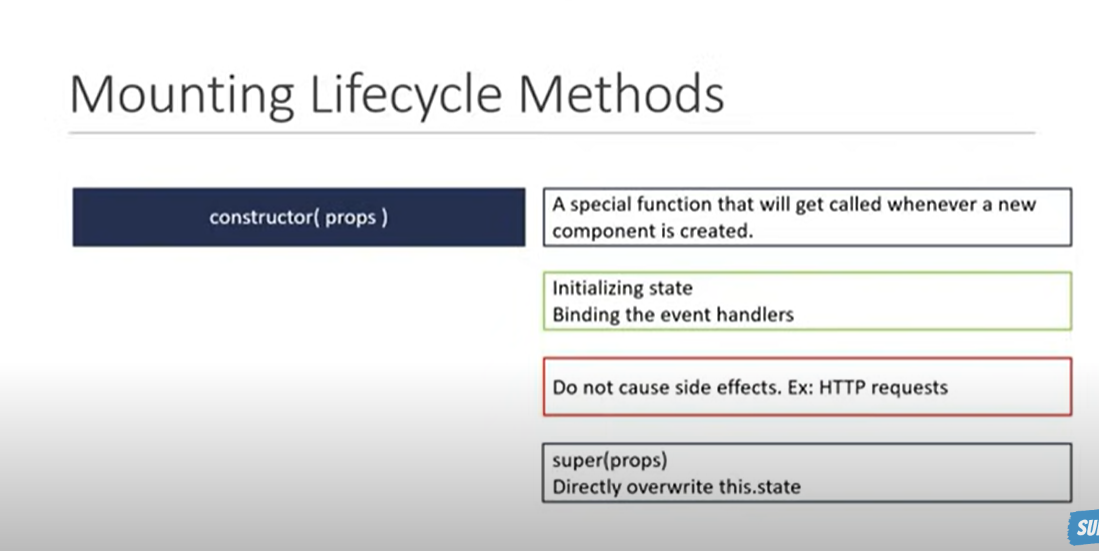


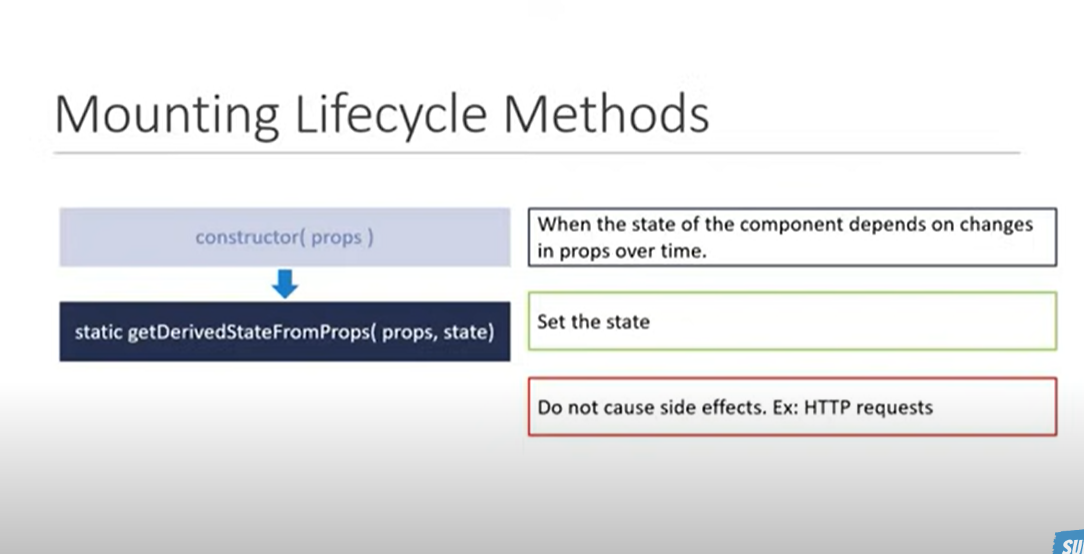
While creating list it is necessary to provide keys attribute which will contain unique identifier to each element in the list.

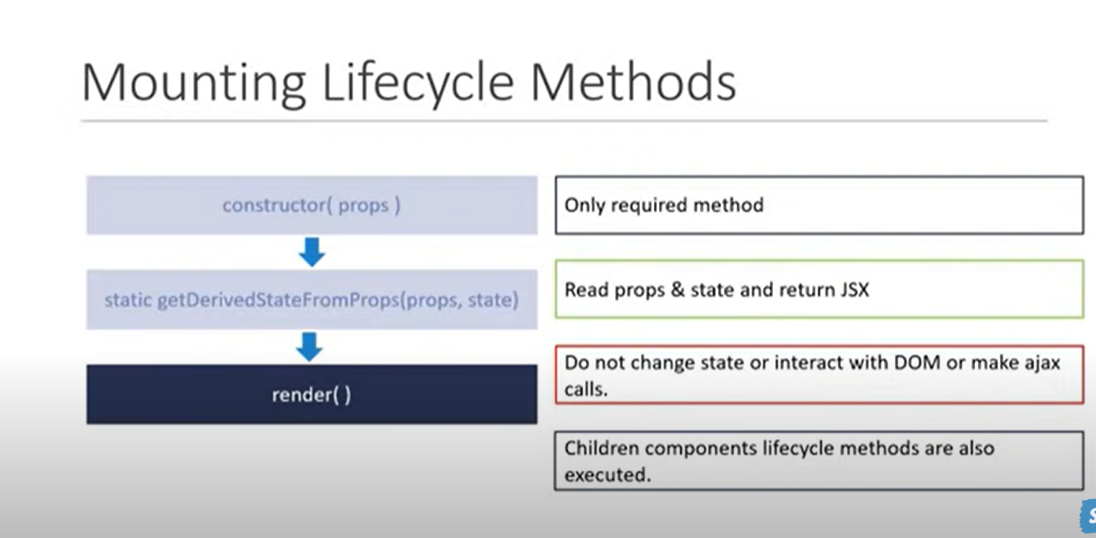


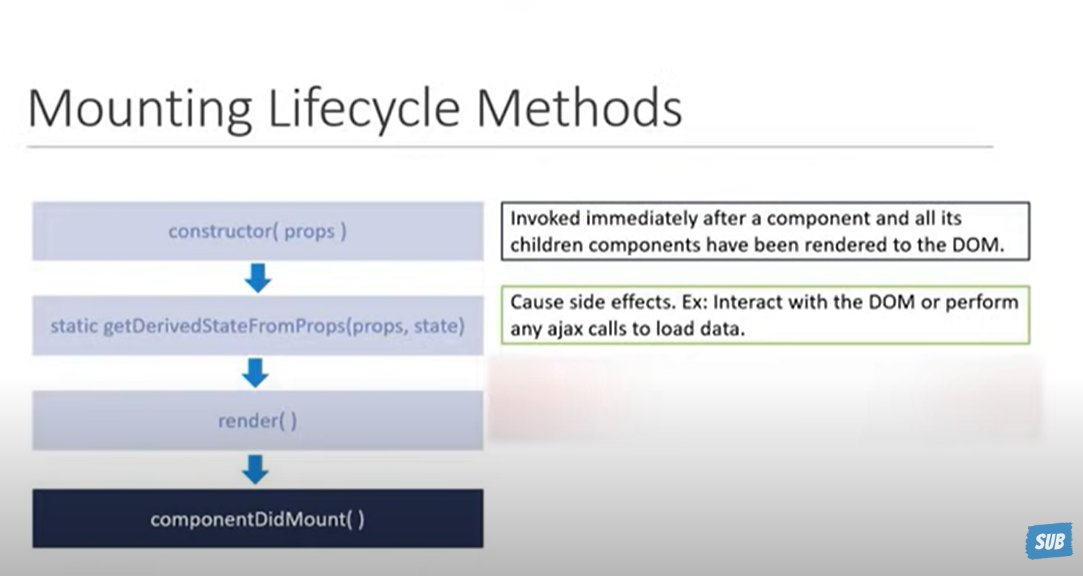






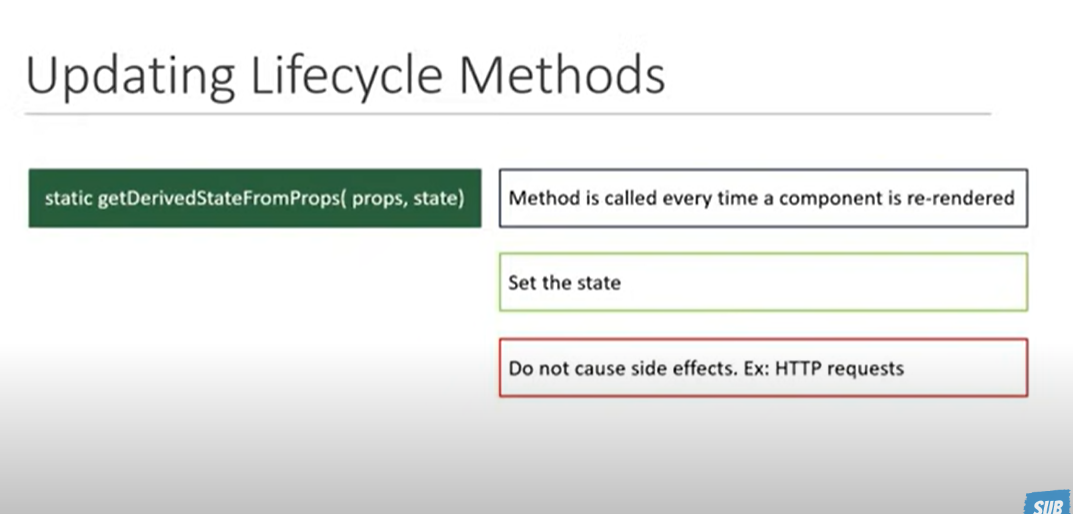


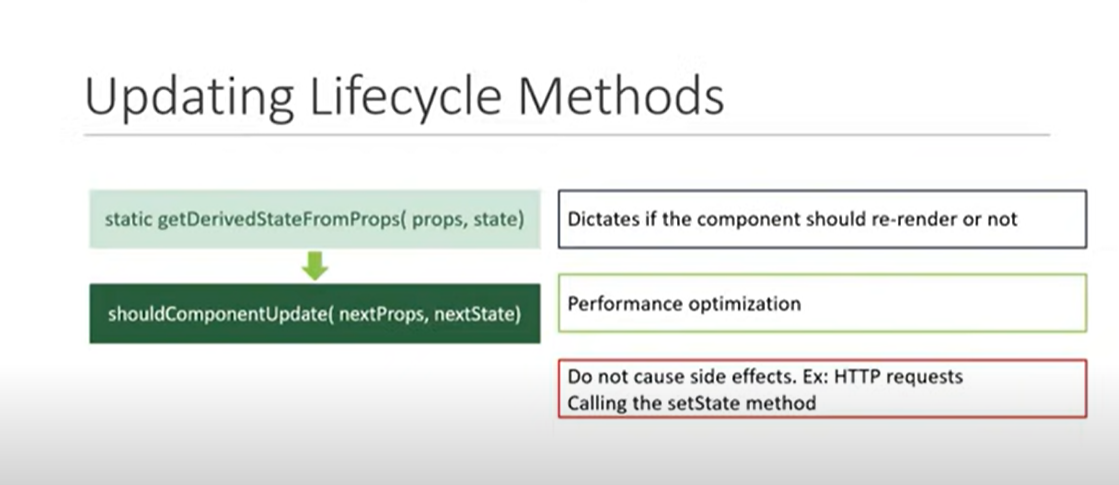


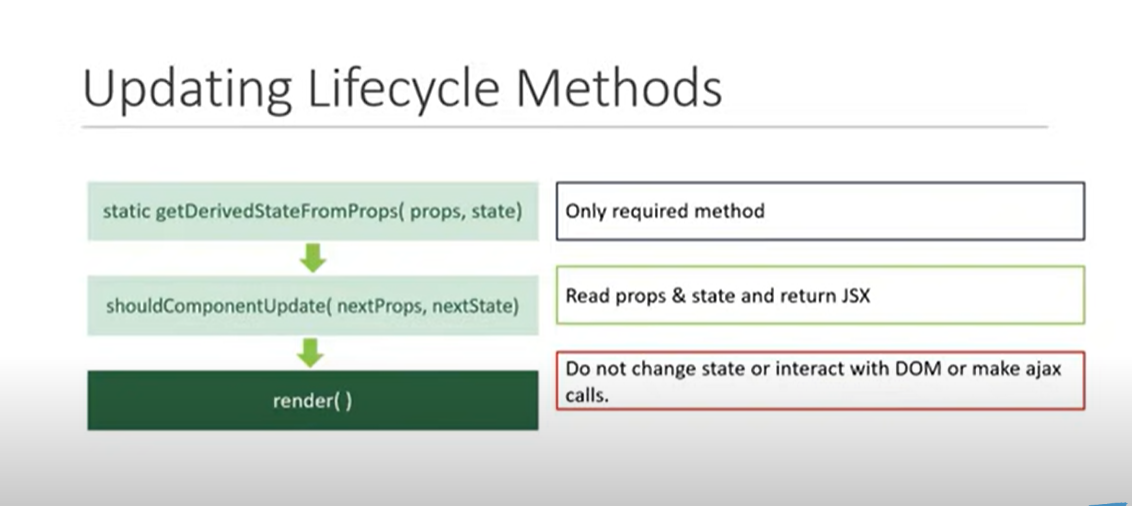


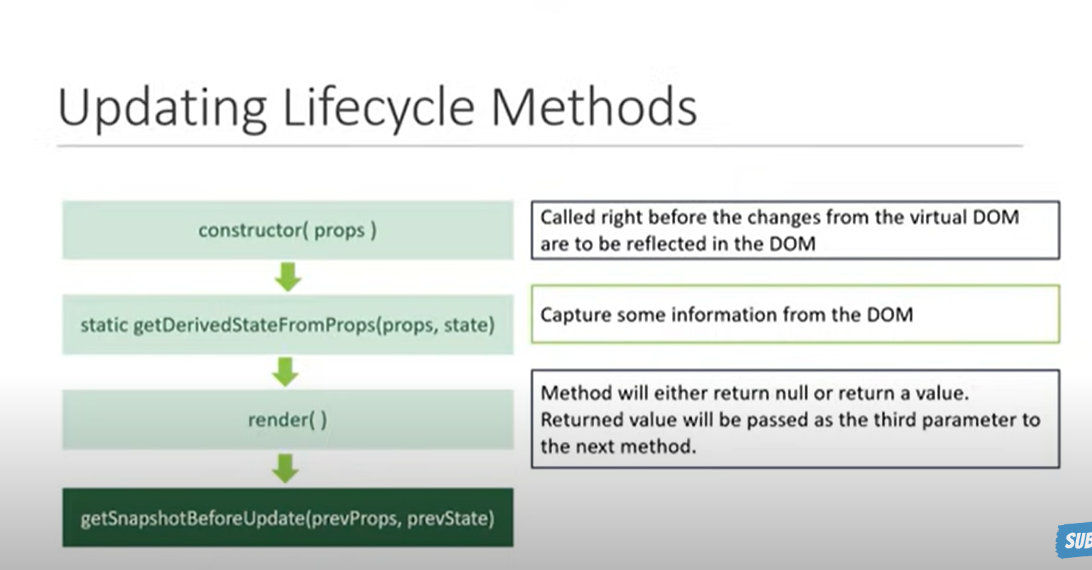
Order of execution of mounting lifecycle methods

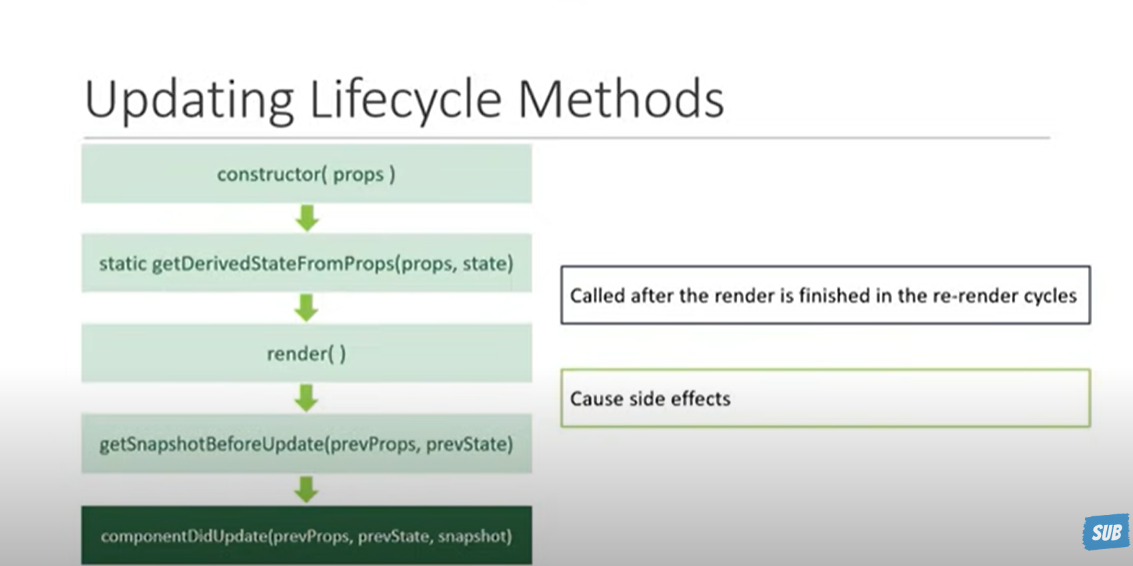


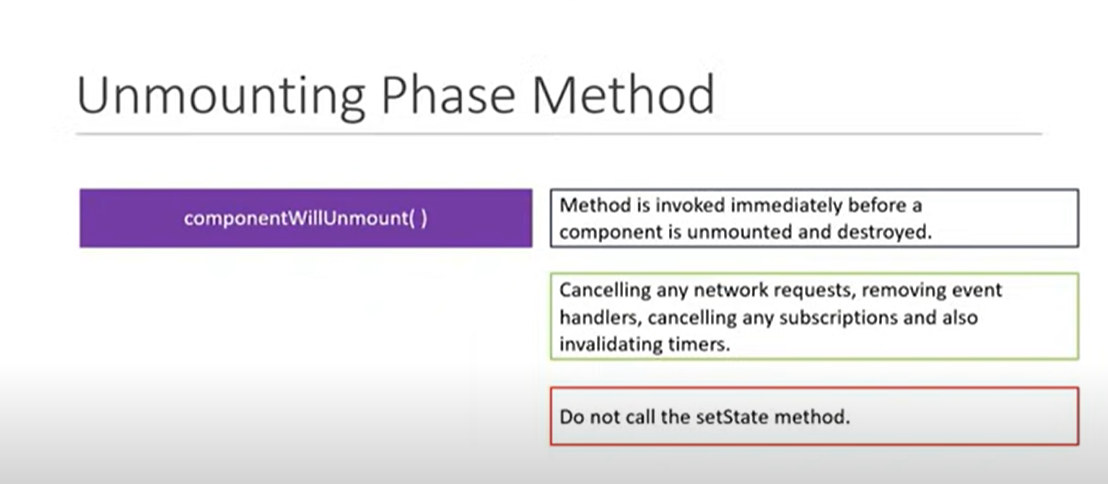


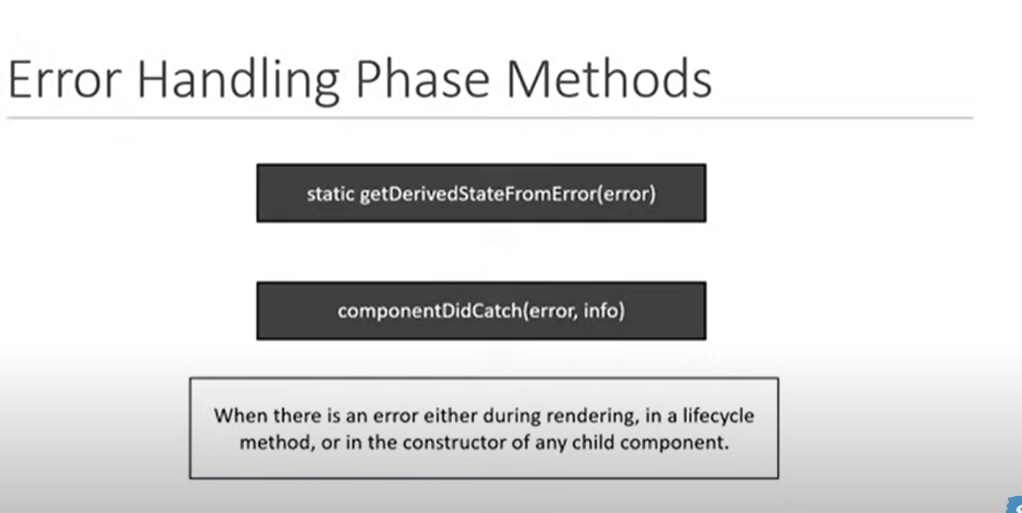






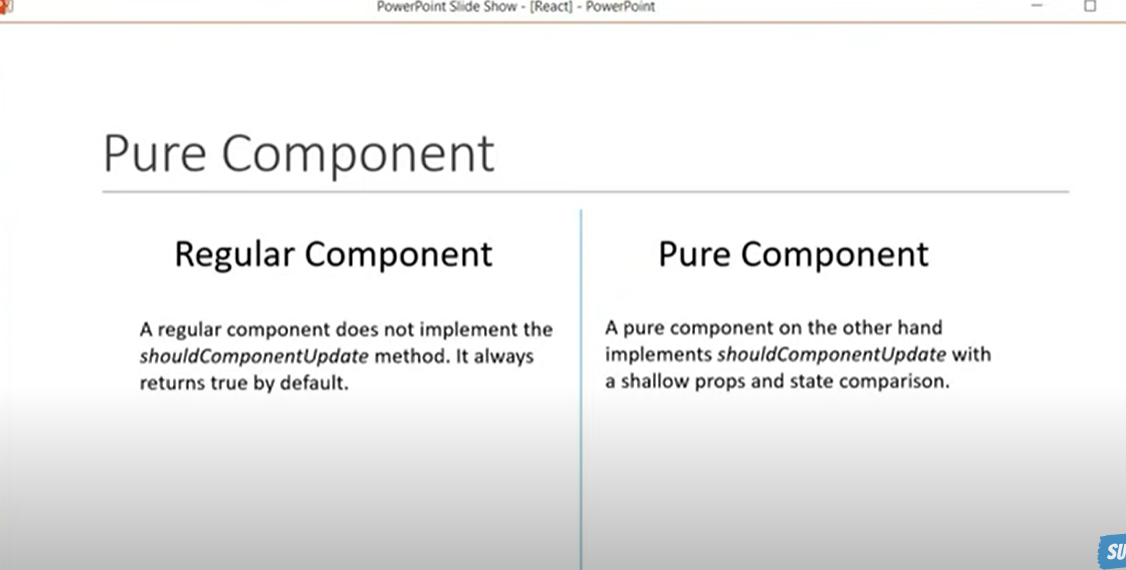


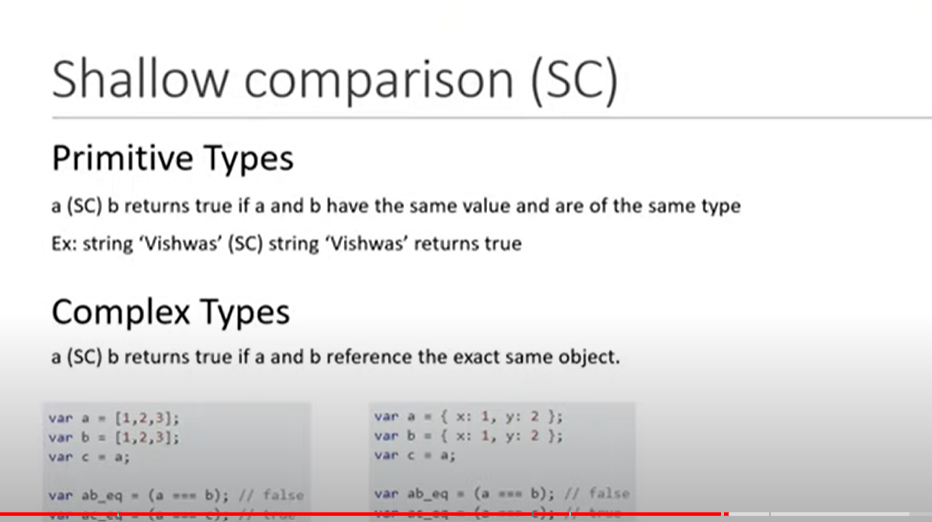




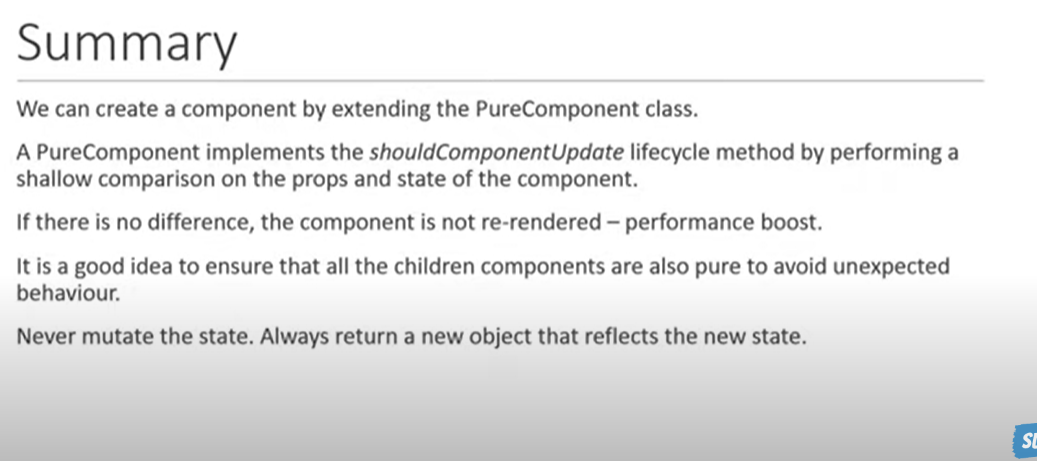
While returning JSX in render method, if multiple elements need to be returned ,then all those elements needs to be enclosed into a single element(in most of the cases it is enclosed under div).

If we don’t need to include that extra element then we may use React.Fragment in such cases.





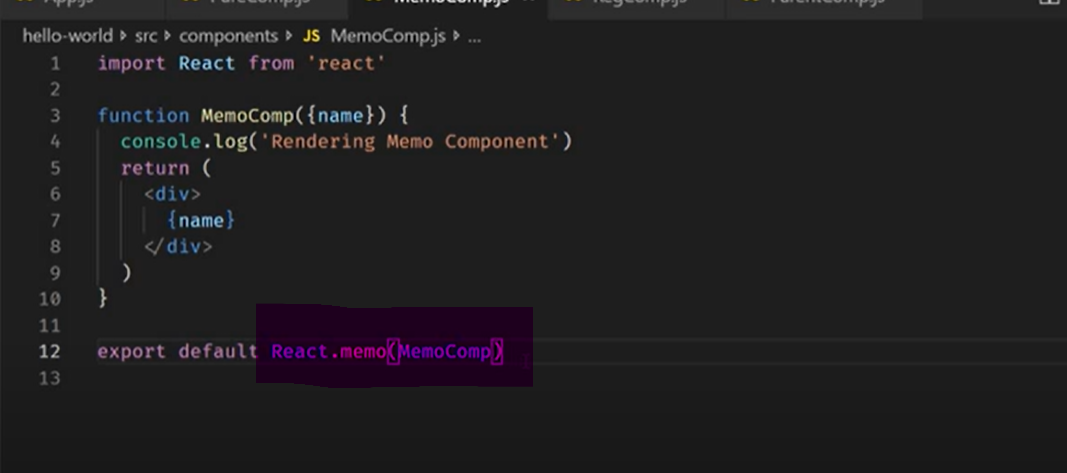




Pure Component should only be used in case of performance issues.

Pure Component is to Component, similarly Memo is to functional component.

Example is given below , everything is similar to functional component, one change is there while exporting functional component is passed to React.memo() function.



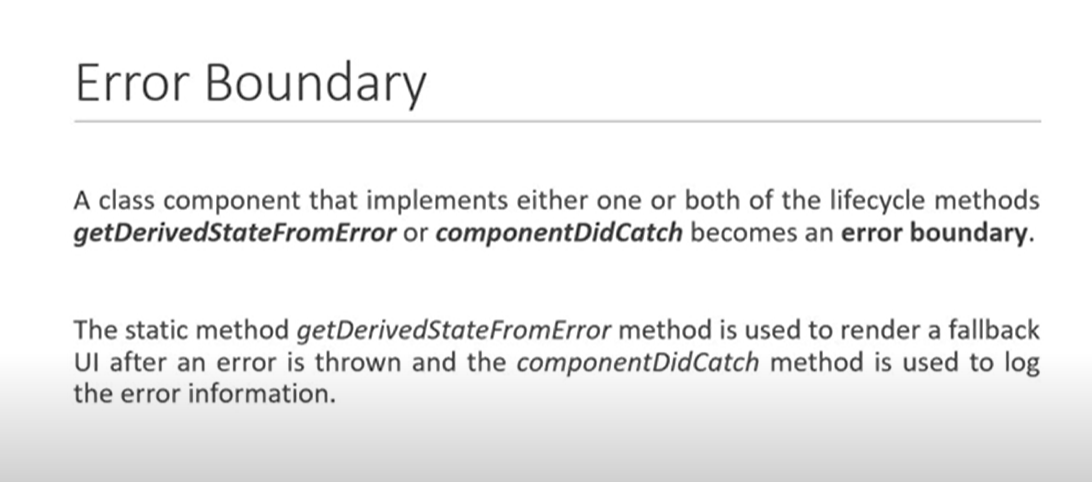
Refs are a function provided by React to access the DOM element and the React element that you might have created on your own. They are used in cases where we want to change the value of a child component, without making use of props and all.

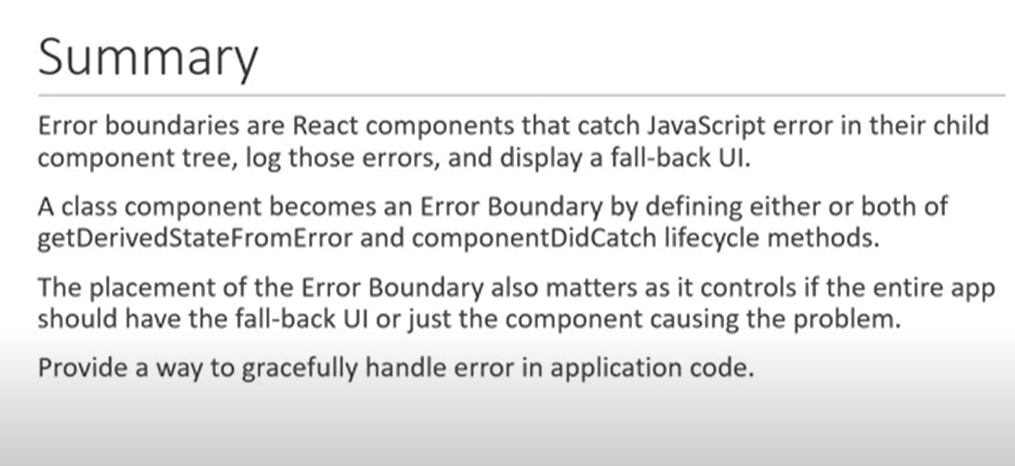
React portals come up with a way to render children into a [DOM](https://www.geeksforgeeks.org/dom-document-object-model/) node that occurs outside the [DOM hierarchy](https://www.geeksforgeeks.org/reactjs-virtual-dom/) of the parent component. The portals were introduced in React 16.0 version.

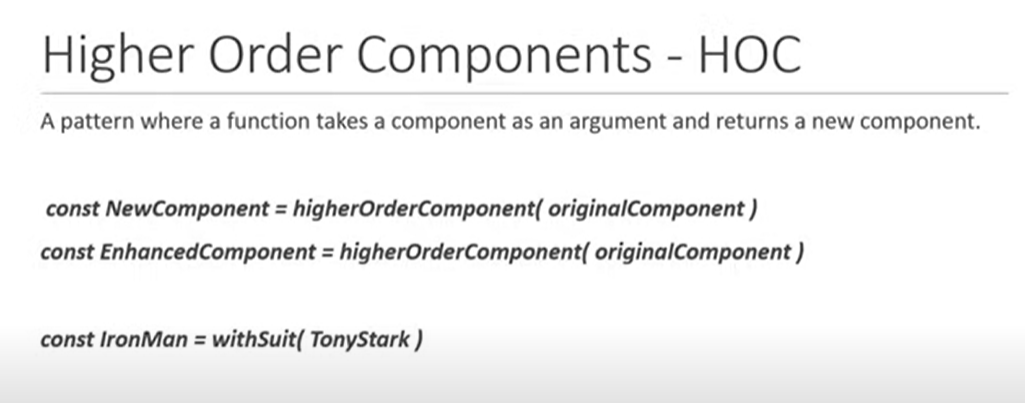
So far we were having one DOM element in the HTML into which we were mounting our react application, i.e., the root element of our index.html in the public folder. Basically, we mount our App component onto our root element. It is almost a convention to have a div element with the id of root to be used as the root DOM element. If you take a look at the browser in the DOM tree every single React component in our application falls under the root element, i.e., inside this statement.

<div id="root"></div>

But React Portals provide us the ability to break out of this dom tree and render a component onto a dom node that is not under this root element. Doing so breaks the convention where a component needs to be rendered as a new element and follow a parent-child hierarchy. They are commonly used in modal dialog boxes, hovercards, loaders, and popup messages.







HOC is a function which takes component as an input and returns an enhanced component

