Assignment 8 - Let's Get Classy

Q1 How do you create Nested Routes react-router-dom configuration?

⇒ In React Router Dom, createBrowserRouter() has children property where it takes an array of Route objects with nested routes on the children property

// Code

```
createBrowserRouter([
    {
        path: "/",
        element: <AppLayout/>,
        errorElement: <Error/>,
        children: [
            {
                path: "/",
                element: <Body/>,
            },
            {
                path: "about",
                element: <About/>,
                children: [{
                    path: "profile",
                    element: <Profile/>
                }]
            }
        ]
    }
])
```

"/" means from the root

It is not "/profile" but "about/profile"

To render this we need to create an outlet in the parent element means inside <About/> or we can directly import Profile component inside Parent

Q2 What is the order of life cycle methods calls in Class Based Components?

⇒ There are 2 phases in React Life Cycle Methods:
Render Phase and Commit Phase

1) Render Phase:
   constructor() → A special method for creating and initializing an object created with a class
   render() → Invoked to examine this.props and this.state in constructor()

React first tries to batch up render phase (render phase of all children as well if present) then commit phase is called.

2) Commit Phase
In commit phase, Mounting, Updating and Unmounting stages are performed

- Mounting → componentDidMount()
  Invoked immediately after a component is inserted into the tree (mounted)

- Updating → componentDidUpdate()
  Called everytime after any update

Q3 Why do we use componentDidMount()?

⇒ This method is not called for the initial render. This is the best lifecycle for DOM and setstate updates. This is also an excellent place to make network requests as long as you compare the current props to previous props.
It is first called in Mounting phase and then everytime in Updating Phase.

Q4 Why do we use componentWillUnmount? Show with example.

⇒ componentWillUnmount() is invoked immediately before a component is unmounted and destroyed. Perform any necessary clean up in this method, such as invalidating timers, cancelling network requests, or cleaning up any subscriptions created in componentDidMount()

You should not call setState() in componentWillUnmount() because the component will never be re-rendered. Once a component instance is unmounted, it will never be mounted again

Eg
Unmouting in Functional Component:

```
useEffect(() => {
    const timer = setInterval(() => {
                    console.log("Namaste React")
                  },1000)

    timer();

    return () => {
                    console.log("useEffect Return")
                    clearInterval(timer)
                 };, []);
```

Unmounting means removing from the page

Q5 Why do we use super (props) in constructor?

⇒ super (props) is used to inherit the properties and access variable of the React parent class when we initialize our component. super () is used inside constructor of a class to derive the parent's all properties inside the class that extended it. If super () is not used, then Reference Error. Must call super constructor in derived classes before accessing 'this' or returning from derived constructor is thrown in the console. The main difference between super () and super (props) is the this.props is undefined in child's constructor in super() but this.props contains the passed props if super(props) is used.

Q6 Why can't we have the callback function of useEffect async?

⇒ useEffect expects it's callback function to return nothing or return a function to return nothing or return a function (cleanup function that is called when the component is unmounted). If we make the callback function as async, it will return a promise and the promise will affect the clean-up function from being called.