Assignment - 8 : Soution

Ques 1: How do you create Nested Routes react-router-dom cofiguration

Solu 1:

- If you want to render children inside children
- steps with example:

1. create profile component

```
const Profile = () => {
    return (
        <div>
            <h2>Profile rendered...</h2>
        </div>
    )
}
export default Profile;
```

2. create children of children route

```
const appRouter = createBrowserRouter([
    {
        path: "/",
        element: <Layout />,
        errorElement: <Error />,
        children: [
            {
                path: "/about",
                element: <About />,
                children: [{
                    path: "profile",
                    element:<Profile/>
                }]
            },
        ],
    }
]);
```

3. Pass outlet component for child component

```
import { Outlet } from "react-router-dom";


const About = () => {
    return (
        <div className="container">
            <h1>About page is coming soon...</h1>
            <Outlet />
        </div>
    )
}
export default About;
```

Note :

- Never use /profile for children of children (nested route). slash means concate with root not the relative path. Instead of /profile, use profile.
- Outlet is replaced by child component. So, you can directly use child component instead of there

Ques 3: What is the order of life cycle method calls in Class Based Components?

Solu 3: Constructor ---> render() ---> componentDidMount()--> render()--componentDidUpdate --> componentWillUnmount()(if we leave the page).

- Class based components are executed in two phases : Render phase & commit phase.
- **Render Phase:**

  1. constructor :

     a. The constructor for a React component is called before it is mounted.

     b. When implementing the constructor for a React.Component subclass, you should call super(props) before any other statement. Otherwise, this.props will be undefined in the constructor, which can lead to bugs.

     c. You should not call setState() in the constructor()

     d. Instead, if your component needs to use local state, assign the initial state to this.state directly in the constructor.

  2. render() :

     a. The render() method is the only required method in a class comp.

- **Commit phase:**

  1. componentDidMount:

     a. componentDidMount() is invoked immediately after a component is mounted

     b. Initialization that requires DOM nodes should go here.

     c. If you need to load data from a remote endpoint, this is a good place to instantiate the network request

  2 . componentDidUpdate() :

     a. componentDidUpdate() is invoked immediately after updating occurs. This method is not called for the initial render.

3. componentWillUnmount() :

    a. This method is called when a component is being removed from the DOM.

- In other way, the whole execution happens in 3 states:
  1. Mounting
     i. Mounting means something needs to be mounted
     ii. For mounting we need initialization, dom rendering, api calling.
     iii. constructor, render and componentDidMound comes under in this phase.
  2. Updating :
     i. render() function is called followed by componentDidUpdate().
  3. Unmounting :
     i. When we leave the page, it gets called.

Ques 4: Why do we use componentDidMount?

Solu 4: It is a Component life cycle method.

- The componentDidMount() method allows us to execute the React code when the component is already placed in the DOM (Document Object Model).
- This method is called during the Mounting phase of the React Life-cycle i.e after the component is rendered.
- All the AJAX requests and the DOM or state updation should be coded in the componentDidMount() method block.
- We can also set up all the major subscriptions here but to avoid any performance issues, always remember to unsubscribe them in the componentWillUnmount() method.

Ques 5: Why do we use componentWillUnmount? Show with example?

Solu 5:

- The componentWillUnmount() method allows us to execute the React code when the component gets destroyed or unmounted from the DOM (Document Object Model).

- This method is called during the Unmounting phase of the React Life-cycle i.e before the component gets unmounted.
- All the cleanups such as invalidating timers, canceling network requests, or cleaning up any subscriptions that were created in componentDidMount() should be coded in the componentWillUnmount() method block.
- Tip: Never call setState() in componentWillUnmount() method.
- Eg:

```
componentDidMount(){
    setInterval(()=>{
        this.tick()
    },1000);
}
```

- It will call componentDidMount() every second.
- cons:
    1. If you will leave the page, it will still gets called.
    2. This is the problem of SPA.
    3. Because it's not reloading of refreshing.
    4. Performance loss.
    5. Needs to be cleaned with the help of componentWillUnmount()
- How to clean :

6. Assign the funtion to the variable.

```
componentDidMount(){
        this.timer=setInterval(()=>{
    console.log("hello")
        },1000);
}
```

2. use componentWillUnmount().

```
componentWillUnmount() {
    clearInterval(this.timer);
}
```

- Component will looks like below:

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }
  componentDidMount() {
    this.timer = setInterval(
      () => this.tick(),
      1000
    );
  }
  componentWillUnmount() {
    clearInterval(this.timer);
  }
  tick() {
    this.setState({
      date: new Date()
    });
    }

    render() {

        return (

            <div>

                <h1>Hello, world!</h1>

                <h2>It is {this.state.date.toLocaleTimeString()}.</h2>

            </div>

        );

    }

}

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Clock />);
```

Ques 6: (Research) Why do we use super(props) in constructor?

Solu 6:

- Super() function is to call the constructor of the parent class.
- It is used when we need to access a few variables in the parent class.
- If super() is not used, then it will throw error in the console. Reference Error : Must call super constructor in derived classes before accessing 'this' or returning from derived constructor
- When you try to use props passed on parent to child component in child component using this.props.name, it will still work without super(props). Only super() is also enought for accessing props in render method.
- 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.

Ques 7: (Research) Why can't we have the callback function of useEffect async?

Solu 7:

- Because React’s useEffect hook expects a cleanup function returned from it which is called when the component unmounts.
- Using an async function here will cause a bug as the cleanup function will never get called.