Day-21 Morning Assessment

Code Splitting y Route:

1. Code splitting is breaking the code into smaller chunks and use it when necessary. By using code splitting we can increase the code efficiency and load only whats needed.
2. import {Suspense, lazy} from “react”;

const About = lazy(() => import(“.About”));

<Suspense fallback={<div>loading…</div>}>

<Route path=”/about” element={<About />} />

</Suspense>

1. The fallback prop in Suspense specifies what to display while the lazy-loaded component is being fetched.
2. It reduces the initial JS file so, the browser downloades and parses less code before showing the page.
3. So, by Route, loads only when the route is visited and by the component, loads specific components on demand.
4. If a dynamically imported route fails to load, it throws an error. You can handle with an Error Boundary or retry logic.
5. Webpack creats separate chunkswith IDs or names based on dynamic import comments or default naming rules.
6. Without webpackChunkName, webpack names chunks with numeric IDs like 0.js, 1.js etc.
7. You can use multiple React.lazy calls and renders them inside the same Suspense components.
8. Yes, you can lazy-load nested route componentsby applying React.lazy and Suspense in the nested <Route> elements.

Webpack Bundle Analyzer:

1. It’s a tool that visualizes the webpack output files,helping you understand whats inside your bundle.
2. npm install –save-dev webpack-bundle-analyzer

in webpack.config.js:

const {BundleAnalyzerPlugin} = require(“webpack-bundle- analyzer”):

plugins: [new BundleAnalyzerPlugin()]

1. It shows a visual treemap of all dependencies, chunk sizes and how code is split.
2. Look for the largest boxes in the treemap-those represent the biggest dependencies.
3. Use tree shaking, dynamic imports, lighter alternatives, or code splitting to reduce size.
4. Development: Run with dev config for real-time updates

Production: Run after building to analyze the production bundle.

1. Static mode: generates HTML file report.

Server mode: starts a server and shows the report live.

1. Use Webpack’s exclude option in the plugin configuration to skip certain packages.
2. Tree shaking removes unused exports after analysis to reduce bundle sizes.
3. SplitChunks break code into separate chuncks, and analysis shows how these chunks are sized and shared.

State Lifting:

1. Lifting State up means moving state from child components to their nearest common ancestor so they can share it.
2. Two siblings may need shared state so both can access and update the same data.
3. Pass a callback from the parent as a prop to the child, and have the child call it to send data up.
4. Too much lifted state can only make the parent overly complex and cause un-necessary re-renders.
5. By storing shared state higher up, fewer components need to pass props down multiple levels.
6. Ex: parent holds form data,input fields are separate child components but update the same parent state.
7. Use React.memo and useCallBack to prevent re-renders of components that don’t depend on changed state.
8. Combine lifted state with react contextto avoid prop drilling across many levels.
9. useCallBack memorizes the callback functions passed to children, avoiding recreation on each render.
10. Keep the parent state synced with value and onChange props for each input to maintain control.