REACT LAZY

&

SUSPENSE

➡ React has a concept of Components Based Architecture which means that instead of building our application as a whole we split our application into multiple reusable components

➡ Now, when we do the production build at that time webpack (Under the hood) will bundle all of the components and it will generate "bundle.js" file

➡ On initial request to any React applications the server sends back a bare minimum HTML file which links to the "bundle.js" file

➡ If our application is very huge then the Javascript bundle will take a long time to download & execute the script file which in turn leads to performance loss

➡ Now, what we can do to avoid this ? Well, you guessed it right this time as well ;)) By using Code Splitting techniques

➡ React Lazy & Suspense are the preferred way to do Code Splitting in React applications

➡ React.lazy takes a function that must call a dynamic import(). The lazy component should then be rendered inside a Suspense component, which allows us to show some fallback content (such as a loading indicator) while we're waiting for the lazy component to load

➡ Now, instead of generating one massive "bundle.js" file webpack will generate components based chunk files which are smaller in size

➡ Let's assume we have a "Profile" component and we have applied code splitting to that component. Then, whenever user visits the "Profile" page (Mounting of the Profile component) at that time React will dynamically import that component

➡ It simply means that the code for the "Profile" component is not included in the initial "bundle.js" file that gets sent to the client. Instead, a seperate chunk file is created for the Profile component which only gets downloaded when user visits that component. I hope that makes sense :)

Syntax ⬇⬇

```
const LazyProfileComponent =
React.lazy(() => import
("./Profile"))
```

DO YOU FIND THIS POST HELPFUL THEN PLEASE DO SHARE THIS POST WITH YOUR CONNECTIONS ;))