**Theory Chapter 09**

* **Chunking**
* **Code Splitting**
* **Dynamic Bundling**
* **Lazy Loading**
* **On Demand Loading**
* **Dynamic Import**

**Q: when and why do we need lazy()?**

**A: NOTE:** all the react app will be bundled into a single file so that our entire app can load at once in the browser. These bundling are done by bundlers like webpack , parcel, Rollup

As our app grows our bundle will also grow. so we need to make sure that our app bundle should not be too large cz it will then result in loading our slow in the browser. So webpack or any other bundler provide a feature {code splitting} so that we can create multiple bundles and load them dynamically in the browser. i.e. on demand bundling

Code-splitting helps to load our components which are needed by the user at that time only ,so this ll help to reduce the initial bundle size and the initial bundle will only contain that amount of code which ll be need just to load our app.

Syntax: const About= React.lazy(()=>import(‘../component/about’))

It will take a callback function which has dynamic import . This will return a promise which will be resolve to a module with a default export containing React component.

**Q: what is suspense and why one need it?**

**A: when** some one is lazy loading a component , it should be done inside a Suspense component so that we can show some loader to the user until our component is being loaded. It accepts a prop fallback which accepts a react component . that react component is shown we one is lazy load a component

When the react component suspends it goes to the closest suspense parent to show the fallback ui. After some time react tries again to load that suspended react component (tree) from the scratch .

**Q: Adv and disadv of using the code splitting pattern>**

**A: ADV**

1. Initial bundler is of small size and our app ll load faster in the browser cz of the small bundle

**DISADV**

1. Code splitting is done only at client side rendering , cannot be done when there is server side rendering

**Q: why we got this error : A Component suspended while responding to synchronous input. This ll cause the UI to be replaced with the loading indicator? To fix updates that suspend should be wrapped with startTransition? How does suspense fix this error?**

**A: when**  user want to see the old ui instead of loading any spinner or loader one has to use startTranstion.

One can use useDeffered to show the pervious result/ui instead of any loader

const [query, setQuery] = useState('');

const deferredQuery = useDeferredValue(query);

const [isPending, startTransition] = useTransition();

function navigate(url) {

startTransition(() => {

setPage(url);

});

}

**NOTE**: Both deferred values and [transitions](https://beta.reactjs.org/reference/react/Suspense#preventing-already-revealed-content-from-hiding) let you avoid showing Suspense fallback in favor of inline indicators. Transitions mark the whole update as non-urgent so they are typically used by frameworks and router libraries for navigation. Deferred values, on the other hand, are mostly useful in application code where you want to mark a part of UI as non-urgent, meaning that it’s allowed to “lag behind” the rest of the UI.