Handwritten Notes on

CHUNKING

"Delivering only what's necessary, when it's necessary"

Save this note

OR

Read it now ⇨

Ashraya K K

For the whole code, parcel creates only one .js file..
In this file, all the ~~fcl~~ code is bundled together.
So, the size of this index.js file is large.
But in production bundle, size of this file should be small.

There would be a 100s of components in a large website like "makemytrip".
Suppose if all these are bundled together in a single index.js file, It will blast. It will make our app very slow.

So, to build a large-scale production ready application, we should do :-

"CHUNKING"

It is also called as :-

→ Code Splitting
→ Dynamic Bundling
→ Lazy Loading.

We cannot bundle everything in our app.

→ On Demand Loading.
→ Dynamic Import

Making a new different bundle in our App.

Let us create "Instamart"

→ create an instamart component.
→ In App.js file, do chunking :-

App.js :-

Instead of importing like this :-

⊛ import Instamart from "./components/Instamart";

Do lazy loading :-

⊛ Const Instamart = lazy(() => import("./components/Instamart"));

So, now the "index.js" file in dist folder won't have code of instamart. It is created as seperate file while loading.
This is called ON-DEMAND LOADING.

"When you are loading your component in demand, react tries to suspend it".

So, when instamart is loaded for the first time, we see an error message on screen.
This is because, instamart file took 27ms to get loaded. But react tries to render it before it get loaded. That's why error.

Solution for this

"Suspense" → We can wrap instamart inside suspense.

App.js :-

```
{ path : "/instamart",
  element : (
        <Suspense>
              <Instamart/>
        </Suspense>
  )
}
```

React now knows that when there is a suspense, what will be loaded.

In the intermediate time, a shimmer should be shown.

So, there is a prop known as "fallback".

So, write :

```
<Suspense fallback = {<Shimmer/>} >
        <Instamart />
</Suspense >
```

NB :-

* Never ever dynamically load your component inside another component.