Appointments

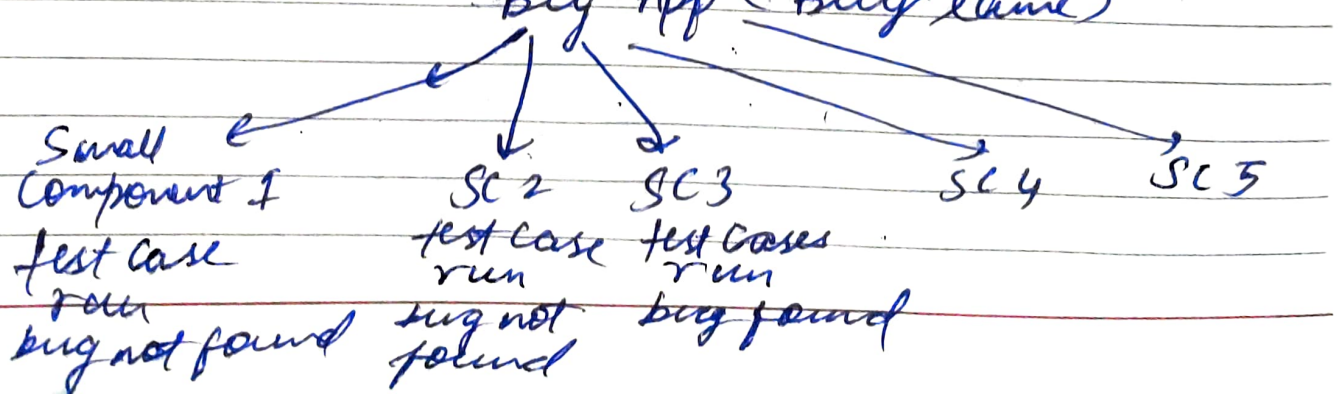
09 ~~## 09~~ Episode-09 - Optimizing-our-app.

10 ## Single Responsibility Principle

11 If you have function, class or ~~piece of~~ code any single identity that should
12 have a single responsibility.

13 Each component should have a single responsibility.
14 Our single responsibility for Header is displaying header onto the page.
15 Our single responsibility for Restaurant card and Restaurant Menu is to display the res card
16 and res menu respectively.

17 This helps in Modularity :- breaking components into small components that help
5 SUNDAY in testing so our big app have a bug that small component or module will catch it in test case so it is easily detectable.



Appointments

So, we change only SC3
This make easy to find bug in our small components and also changes in code which makes it easily detectable and maintainable.

Modularity features:
More ~~reuse~~ reusable
More maintainable
More testable.

Custom Hooks

special
Hooks are functions ~~that~~ provided by react that let function components use React features such as state and lifecycle behaviour.
useState → manage state
useEffect - run side effects
useContent - consume content.

We have a RestaurantMenu component which has two responsibilities :- fetch the data & display it, but we will create a custom hook for fetching the menu data so the only RestaurantMenu is to display it.

Appointments

09 Industery standard for custom Hook
Create that file for hook in utils
10 and always create a separate file
for separate hook. and create a
11 name with 'use' like
'useRestaurantMenu' as react understand
that use means a hook.
12

13 const useRestaurantMenu = (resId) => {
// code
14
return resInfo;
15 }

16 export default useRestaurantMenu.

17 This helps in testing as I have
to check the fetching hook if there
is no data to display and if there
is error in displaying we need to
check RestaurantMenu only.

Online / offline feature

Basics of creating of custom Hook.
Finalizing the contract.
- What we need in the hook?
- What should it return?

Bundler makes bundle all the files of your code in one file.

Appointments

09 We will use event listener called 'window: online' event listener. We need to add this
10 only once and we will use useEffect for adding only once.

11
We can simulate offline behaviour via
12 browser by going into N/w tab and ~~then~~ make the connection to offline.

13
=) Optimizing the app.

14
Parcel is a bundler which bundles all
15 the files of your code in one js file.
=) Cons of making 1 JS file.
16 If we bundle a large project into one JS file, the size of the file will increase
17 a lot.
=) How to optimize this?
So, we have 2 problems
- We do not want all the files to load on browser.
- We do not want that all the files to load in a single file.

So, we will make smaller bundles of this file. This process is known as Dynamic bundling or code splitting or chunking or lazy loading or on-demand loading, etc.

2025

Appointments

09 How to & when to make smaller bundles:

10 Logical separation of bundles so that
it has enough code for a single big
11 feature of website.
For eg:- makemytrip can make bundle of
12 flight ticket, another for hotels,
another for trains, etc. so that we
13 have small application in a single
application.

14

Just assume we have also Grocery
15 which has many components inside it
so, we make two bundle one for
16 Grocery & one for food delivery.

17 We will import our grocery via lazy loading.
Initially our code will not have grocery
code when we go to grocery page
then only we load grocery page.

We will import grocery like this for lazy loading.

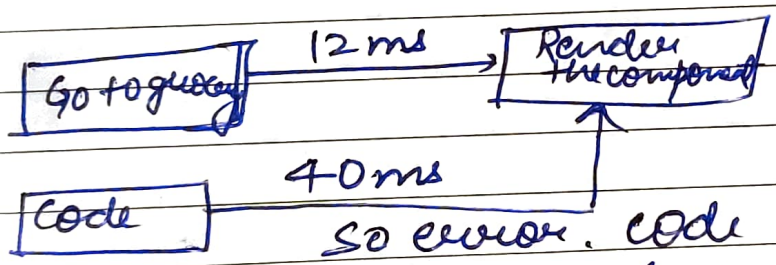
```
import {lazy} from "react"

cons Grocery = lazy(() => import("./components/
                                  grocery"));
```

Appointments

When we look into N/w tab now the grocery will have its own js file & other bundle file will be different

So adding only that line will throw an error as react will try to load the component but the code is not there so it will throw error. as the file will load but it doesn't have the code



so error. code needs more time but react is very fast so it load the component before the code came. So, we will use 'Suspense' keyword here

```
import {suspense} from "react"

<Suspense> <Grocery/> </Suspense/>
        ↓
fallback={<Shimmer/>}
```

So, until the code comes react needs to load something, we pass it in fallback and we can pass any JSX piece in it.

11 SATURDAY

Week 41 · Day (284-081)

OCTOBER

O	M	T	W	T	F	S	S	M	T	W	T	F	S	S
C			01	02	03	04	05	06	07	08	09	10	11	12
T	13	14	15	16	17	18	19	20	21	22	23	24	25	26
25	27	28	29	30	31									

Appointments

09 Suspense is used to manage asynchronous
data fetching and code-splitting allowing
you to display loading indicators
10 while the data load or code ~~is~~ being
fetched. When suspense is encountered
11 by react it knows there might be
a delay in rendering & it can handle
12 that situation gracefully.

13

14

15

16

17

12 SUNDAY

2025