L-9

optimising

- modularity means → break down code into different small modules so that code become more maintiable & ~~sustainable~~ testable.
- Singl Responsibility principle → Each of the component should have single responsibility.

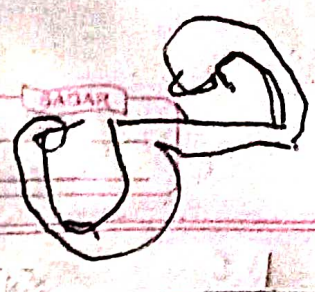
- How modularity helps ?
- it helps ~~in~~ maintainance & in testing of code as because if code is for specific function & in specific component then as to check or catch bug we have to ~~write~~ test for each component, so ultimately it helps to catch bug ~~& solve it~~.

- distribute code in to different component & keeping it in modules

- it helps in scalable, maintainable and testable.

→ Custom hooks

- Creating custom hook is not mandatory but it make your code more reusable & modular.



→ § Convention

↳ 'use' - keyword for filename with the same name of hooks
↳ it is not mandatory

Why we uses small letters keyword use in naming of hooks:
↳ So, React understand that it is hook by (it identify by small letter use).

Hook is like utility function which is just helper function

→ To create custom hooks

1st of all

↳ finialise the contract :- what is the input of hook and output of hook
↳ Do we need any info.
↳ Caller → who is going to be called

So:

↳ - body fetch data logic create custom hooks

↳ can turn off net for that website by using browser - Go to browser
Go to Network & disable cache
Click on No throttling option & select offline

178

Page No.
Date

11.c JS file size increases how many components it hold as it need to optimize for large scale app. - it need to fix otherwise it takes lots of time to load.
- break in to smaller smaller files/logic chunks, not too many small files this process is called chunking, & code splitting / dynamic ~~loading~~ / bundling, dynamic import.

~~o~~ we should not put load on single bundle ~~of this~~ JS file should not ~~take~~ to so heavy that it takes lot of time to load.

but how to chunks ~~means~~ - it depend on your app scenario if your app sell food & it has also a sell grocery so, it is best to chunk it.

Now To have seperate bundles for Grocery we will not import ~~bundles~~ Grocery component in App.js but we use lazy loading

lazy loading :- means Initially our app does not ~~have~~ load Grocery code it loads only when we visit Grocery link. it is also called on demand loading.

we say -

Page No.
Date

Const Grocery = lazy(() => import('./Grocery'));

here lazy() is function which is given by React

This import is not the import we generally use for import file ; it is Callback function & it takes the path of Grocery/component.

and also you have to import {lazy} from react

- This one line of code lots of things

Now if you go the grocery link & in Inspect go to Network & click on JS you will see different Grocery/component.js file that's power. & now it has own bundle.

here you see file but it has throw an error as because it takes their some time to load React Script as code is not there so it throw an error

So, to handle this error we will use Suspense!

Page No. ______
Date ______

Suspense = it is Component Comes from React

We have to just import Suspense &
& wrap the Grocery with Suspense in app.js
~~<Suspense> <Grocery> </Suspense>~~

So, meanwhile when it loads Grocery
for that React can show something
for this we use:

<Suspense fallback = { <h1> loading </h1> } > <Grocery /> </Suspense>

JSX or
~~component~~
component here,
however can also be
<Shimmer />

This how we distribute application into
smaller chunks. ~~This is called dynamic loaded~~

Assignment - 9

Q) When & why do we need lazy()

lazy() function is used for lazy loading components which means component is loaded only when it is actually needed.

lazy/use when you have large ~~or~~ or infrequently used component that you don't want to include in main bundle initially, ~~you~~ Instead you can split it in to separate chunk that loaded asynchronously when its needed. This can be especially useful for optimizing loading time of your application, particularly for larger application.

To use lazy() - import from react & wrap
~~detailed in previous page~~

on main page — Const Grocery = lazy(()=>import("path"))

Q) What is Suspense?
→ it is component given by React, it is used to show a fallback UI such as loading spinner, while the component is being loaded.

~~Syn~~ 8 <Suspense fallback={<h1>loading</h1>}> <Grocery/> </Suspense>

Remember:- we have to wrap ~~[illegible]~~ Lazy by Suspense.

Page No.
Date

2) why we got this error: A component suspended while responding to synchronous input. This will cause the UI to be replaced with loading indicator. To fix updating that suspend should wrapped with start Transition? How does suspension fix this error?

A) As After lazy() function execution browser takes time to load that page meanwhile React will found no code. So it will throw an error. So to fix this we use suspension. Suspension basically if the lazy-loaded component has not finished loading yet. React will suspend the rendering process at that point. while component is suspended, suspense component define what should be displayed in meantime such as loading spinners. once lazy-loaded component finished loading React resumes rendering process & replace the placeholder content with actual component -

Shorty → <Suspense fallback={<h1>loading...</h1>}> </Suspense>
placeholder

what suspense is doing. that it replace placeholder by Suspense actual content while lazy loading & display the fallback value and after lazy loading complete it replace placeholder content with actual content

Page No.
Date

Q. Advantages and disadvantages of using this code splitting pattern

Advantage:-

- Improved performance:- Code splitting allows you to load only essential parts of your application upfront, reducing initial bundle size. This improves initial loading time.
- Faster Time to Interactive:- By loading components lazily, you can prioritize the loading of critical component first.

Disadvantage:-

Increased Complexity:- Implementing code splitting and lazy-loading can introduce additional complexity to your application as you need to consider how and when to split your code, manage loading state and handle error scenarios. This complexity can add to the development time and require a good understanding of underlying concepts.

Page No.
Date

⇒ When do we and why do we need Suspension

During lazy loading we need Suspension as if Suspense actual content while loading it display fallback values and after lazy loading completely replace placeholder content with actual content. This enhance the user experience and also help to render lazy load Components without any error.