Ep·09 Optimizing Our App

Date:

↳ modularity

Breaking down code into small small modules.

Because of it, code becomes more maintainable, testable.

When we have single responsibility for single component we can write test cases for only that component

If we have bug in app, we easily able to catch it with the help of test cases.

Distributing our code into smaller smaller pieces and keeping it modular makes our code testable & maintainable

If we write code in modular fashion and follow single responsibility principle we get another one feature is reusability.

1) keep component as light as possible. Code should be redable

↳ Custom Hooks

- Creating own hooks.
- We can abstract some responsibility of component and extract inside a hook, so that our hook and our component becomes more modular. and redable
- We can optimize our code using custom hooks.
- e.g fetching data from an api, we can create custom hook for it component need to only focus upon the displaying data

• why custom hook?

Creating custom hook is not a mandatory thing.

But it is good thing to make our code more modular, reusable, readable

- Always create hooks inside utils and prefered to create separate files for separate hook

↳ checking user's onlineStatus feature

- How to think to write custom hooks ?

1) Finalise the contract.
   ↳ what is the input of the hook.
   ↳ what is the output of the hook

We can build this features using Custom hooks.

↳ chunking / code splitting
Dynamic Bundling / lazy loading
ondemand loading / dynamic import

Logically splitting our website code into smaller bundles, so don't put a load on one bundle, which takes a lot of time to get into browser

Loading files in the browser when it's required. That's why also known as lazy loading.

- lazy() :

Function given by 'react', comes as named import

lazy() Function takes a callback function.

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

. Suspense :

It is a component comes from 'react' library.

e.g `<Suspense fallback = { <h1> Loading .... </h1> } > <Grocery /> </Suspense>`

fallback is something like shimmer or placeholder, to show while fetching the data for Grocery component