Single responsibility function / Modular code

98th Episode
24/03/24

↳ Every function should have a single responsibility.

ex- Component <RestaurantMenu>
→ So this function should have single responsibility here it is of show the restaurant Menu.

→ Break down you code is smaller sections.

Advantages

↳ Easy to debug.
→ Modular code
→ Code is reusable

Custom Hooks

↳ So custom hooks just are helper functions that are written in utility. Make the code looks simpler.

① creating custom hook in RestaurantM component.
② create hook
③ export it and imp
④ use in restaurant M

Creating Hook

→ ① Start the name with **use** So that React understand it is a Hook.

ex- useRestaurantMenu.js

useRestaurantMenu.js
import

```
const useRestaurantMenu = (resId) => {
  [same code]
```

Creating useOlineStatus Hook

① We will use online event listner from window object.
② using a use effect hookk for window.addEventListner
③ vice-versa for online
④ return online status

```
window.addEventListner("offline", () => {
    setOnlineStatus(false);
});
```

(5) In body return a <H1> Looks like you are offline </H1> by using the useOnlin Hook.

(6) Add a Green or Red Dot on Header

→ a) Import the hook in header

const onlineStatus = useOnlineStatus();

 Online Status : {onlineStatus ? "✅" : "🔴"}

emoji

Optimizing Our App by breaking JS files smaller bundeles / Chunking / code spliting.

→ Dynamic Bundling / lazy loading / On demand loading

~~Import~~ lazy, Suspense from react

⇒ because if only on JS file it will be of very large file, slow to load so we make logical seprations of our code.

→ b)
```
import {lazy} from "react"
const Grocery = lazy(() => import("./components/grocery"))
```

Grocery: component

import(): a function

→ this code only loads when use the component

```
import {Suspense} from react;
```

c) Wrap ~~load~~ the component in

```
<suspence> ... </suspence>
```

As react is very fast so it will try to use this code before even the code is loaded in browser so ~~those~~ use the above tag to Wrap

ex:

```
{
  path : "/grocery",
  element : (
    <Suspence fallback = {<h1>Loading</h1>} >
      <Grocery />
    </Suspence>
  )
}
```

→ Now while it loads that JS file till that time we need to show something so we use this fall back function for that.

* This lazy loading / dynamic loading is very powerfull it makes our application very fast.

(11) done lazy loading for Instamojo Component.

→ You can see the different JS bundles in Browser / inspect → Network → JS