**Custom Hook 🔥**

1. **Why to create own Hook?**

Reusability, Readability, Maintainability — easy to debug, Testable

**2. How to use functions as a Hook?**

Hook is a normal javascript function.

Extract some function

eg) utils -> useRestaurant.js

import { useState } from "react";  
  
const useRestaurant = (resId) => {  
 const [restaurant, setRestauraunt] = useState(null);  
// get data from api  
useEffect(() => {  
 getRestaurantInfo();  
 }, []);  
  
 async function getRestaurantInfo() {  
 const data = await fetch(  
 "https://www.swiggy.com/dapi/menu/v4/full?lat=12.9351929&lng=77.62448069999999&menuId=" +  
 resId  
 );  
 const json = await data.json();  
 console.log(json.data);  
 setRestauraunt(json.data);  
 }  
  
 return restaurant  
  
 // return restaurant data  
};  
  
export default useRestaurant;

import useRestaurant from "../utils/useRestaurant";  
import Shimmer from "./Shimmer";  
  
const RestaurantMenu = () => {  
 // how to read a dynamic URL params  
 const { resId } = useParams();  
 // Use proper names  
 // const [restaurant, setRestauraunt] = useState(null);  
  
 const restaurant = useRestaurant(resId)  
 // useEffect(() => {  
 // getRestaurantInfo();  
 // }, []);  
  
 // async function getRestaurantInfo() {  
 // const data = await fetch(  
 // "https://www.swiggy.com/dapi/menu/v4/full?lat=12.9351929&lng=77.62448069999999&menuId=" +  
 // resId  
 // );  
 // const json = await data.json();  
 // console.log(json.data);  
 // setRestauraunt(json.data);  
 // }

**3. Shared Folder for Shared Function**

* Create a utils/ common/ helper/ shared folder

//utils -> helper.js  
  
export function filterData(searchText, restaurants) {  
 const filterData = restaurants.filter((restaurant) =>  
 restaurant?.data?.name?.toLowerCase()?.includes(searchText.toLowerCase())  
 );  
   
 return filterData;  
 }  
  
  
//other components  
import { filterData } from "../utils/helper";

**4. How to Create a own Hook? To check if online or offline**

Window: online event → js function

import { useEffect, useState } from "react";  
  
const useOnline = () => {  
  
 const [isOnline, setIsOnline] = useState(true)  
  
 useEffect(()=>{  
 window.addEventListener('online', () => {  
 setIsOnline(true);  
 }  
 );  
 window.addEventListener('offline', () => {  
 setIsOnline(false)  
 }  
 );  
 },[]);  
  
 return isOnline // true or false  
  
}  
  
export default useOnline;

import useOnline from "../utils/useOnline";  
  
const isOnline = useOnline();  
if(!isOnline) {  
 return <h1> 🔴 Offline please check internet connection !</h1>  
}

**5. Lazy loading**

Building very Big App. As React is a SPA instead of calling all api. Use Lazy loading to Keep different bundles of app in different chunks eg) in Amazon for dress section call dress api, for electronics section call electronics api.

fallback -> loading, shimmer, spinner

The React.lazy function **lets you render a dynamic import as a regular component**.

import {lazy, Suspense } from "react"  
  
  
const Instamart = lazy (() =>import("./components/Instamart"));  
//Upon on demand loading -> upon render -> suspend loading  
  
  
{  
path:"/instamart",  
element: {  
<suspense fallback={<Shimmer/>}>  
<Instamart/>  
</suspense>  
}  
}

**6. Do and Don’t in Lazy Loading**

Lazy load on the top. Right after import.

Don’t call lazy load inside the comp

