* **npx create-react-app netflix-gpt**

**# adding / Installing tailwindcss in project**

* + tailwindcss website> get started > framework guides > create react app
  + **Install Tailwind CSS :-** Install **tailwindcss** via npm, and then run the init command to generate your **tailwind.config.js** file.
    - **npm install -D tailwindcss**
    - **npx tailwindcss init**
  + **Configure your template paths :-** Add the paths to all of your template files in your **tailwind.config.js** file.
  + **Add the Tailwind directives to your CSS :-** Add the **@tailwind** directives for each of Tailwind’s layers to your **./src/index.css** file.
  + <https://tailwindcss.com/docs/guides/create-react-app>

**#Router Configuration**

* **npm i -D react-router-dom**
* appRouter = createBrowserRouter > path and element
* import RouterProvider 🡪 <RouterProvider router={appRouter}/>

**#Login Page / Sign-up Page**

* **Check Formik form validation also for big data validation**

**#Form Validation**

* **Regex for email & password validation**
* How can I validate an email address using a regular expression?
* <https://saturncloud.io/blog/how-can-i-validate-an-email-address-using-a-regular-expression/> 🡨 Email
* <https://regexr.com/3bfsi> 🡨 Password
* There is test function over regex and we want to test our email and password
* Validate User Full name
* 🡺 <https://regexr.com/3f8cm>

**export *const* checkValidData = (email, password) => {**

***// if this regex is pass then it will return TRUE inside it or FALSE inside "isEmailValid"***

***const* isEmailValid = /^([a-zA-Z0-9.\_%-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,})$/.test(email);**

***// if this regex is pass then it will return TRUE inside it or FALSE inside "isPasswordValid"***

***const* isPasswordValid =/^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[a-zA-Z]).{8,}$/.test(password);**

***// return's Error Message***

**if (!isEmailValid) return "Email ID is not valid";**

**if (!isPasswordValid) return "Password is not valid";**

***//if both are valid then return null i.e. No Error***

**return null;**

**};**

**useRef Hook**

* useRef is a React Hook that lets you reference a value that’s not needed for rendering.
* <https://react.dev/reference/react/useRef>
* <https://refine.dev/blog/react-useref-hook-and-ref/#introduction>

**#Authentication | Using Fire-base for Authentication**

* Using Google Fire base for Back-End
* Get Starte 🡺 Add Project 🡺 Project-Name 🡺 Default Account for Firebase
  + Get started by Adding Firebase to your web 🡺 Create a web-project
  + Register app
    - App-nickname / project name
    - ✔️ also set up Firebase Hosting for this app

* + Add Firebase SDK
    - npm install firebase
    - create firebase.js file in utils and paste that i.e. initialize firebase config
      * In your project go-to Project 🡺 overview 🡺 Enable Authentication
      * Get Started 🡺 sign in method 🡺 add sign in option 🡺 enable it
  + Install Firebase CLI
    - * To host your site with Firebase Hosting, you need the Firebase CLI (a command line tool).
      * Run the following [npm](https://www.npmjs.com/) command to install the CLI or update to the latest CLI version.
      * **npm install -g firebase-tools**
  + Deploy to firebase Hosting
    - You can deploy now or [later](https://firebase.google.com/docs/hosting/quickstart?hl=en&authuser=0). To deploy now, open a terminal window, then navigate to or create a root directory for your web app.
    - **Sign in to Google**
      * **firebase login**
    - **initiate your project**

Run this command from your app's root directory:

* + - * **firebase init**
    - **when you’re ready deploy your web app**

Put your static files (e.g., HTML, CSS, JS) in your app's deploy directory (the default is "public").

Then, run this command from your app's root directory:

* + - * **firebase deploy**
* After deploying, view your app at [netflixgpt-d91a2.web.app](https://netflixgpt-d91a2.web.app/)
* Need help? Check out the [Hosting docs](https://firebase.google.com/docs/hosting/quickstart?hl=en&authuser=0)

**#Firebase Documentation**

* <https://firebase.google.com/docs/auth/web/password-auth?hl=en&authuser=0>
* Firebase 🡺 Docs 🡺 Builds 🡺 Authentication – Web 🡺 Password Authentication
  + **Sign-Up** **a user with an email address and password**
    - Choose web Modular API
    - Create a password-based account 🡺 API – createUserWithEmailAndPassword **To create a user with email and password on firebase**
    - *import { getAuth,* ***createUserWithEmailAndPassword*** *} from "firebase/auth";*

**createUserWithEmailAndPassword(**

**auth,**

**email.current.value,**

**password.current.value**

**)**

* + **Sign-in** **a user with an email address and password**
    - *import { getAuth,* ***signInWithEmailAndPassword*** *} from "firebase/auth";*

**signInWithEmailAndPassword(**

**auth,**

**email.current.value,**

**password.current.value**

**)**

**#after building sign-in and signup feature**

* what will do now is try to push this user object we got / that user information we got we will put that info into our Redux Store

**#Redux Store for Sign in & Sign up**

* If the user sign up or a sign in we will got this user object and we will have to keep the user object with us because we need this user object anywhere in our app.
* So what we will do is as soon as the user sign in or sign up We will just add all that data to our redux store i.e. once user sign in or sign up we will add user to our redux store.
* **npm i -D @reduxjs/toolkit**
* **npm i react-redux**
  + utils > appStore.js 🡪 configureStore – reducer collection of different reducers
  + utils > userSlice.js 🡪 createSlice – name, initialState – reducers{add or remove} function
  + **userSlice**
* **import { createSlice } from "@reduxjs/toolkit";**
* ***const* userSlice = createSlice({**
* **name: "user",**
* **initialState: null,**
* **reducers: {**
* **addUser: (state, action) => {**
* **return action.payload**
* **},**
* **removeUser: (state, action) => {**
* **return null**
* **}**
* **}**
* **})**
* **export *const* { addUser, removeUser } = userSlice.actions**
* **export default userSlice.reducer**
  + **appStore**
* import { configureStore } from "@reduxjs/toolkit";
* import userReducer from "./userSlice"
* *const* appStore = configureStore({
* reducer: {
* user: userReducer
* }
* })
* **export default appStore**
  + **now providing Store to our App**
* *function* App() {
* return (
* <Provider *store*={appStore}>
* <Body />
* </Provider>
* );
* }
* Now what we will do next thing is as soon as my user sign in or sign up I want to update my store I want to update my userSlice with that user Information
* I will have to dispatch an action whenever user sign in and sign out, So instead of doing that dispatch again and again here we will use a utility which is given to us by firebase which is known as **onAuthStateChange**
* **Manage Users in Firebase** ~~(not using redux dispatch)~~ - **onAuthStateChange**
* <https://firebase.google.com/docs/auth/web/manage-users?hl=en>
* Firebase 🡺 Docs 🡺 Builds 🡺 Authentication – Web 🡺 Manage User
* Firebase gives us an amazing API that is known as onAuthStateChange this API is called whenever the user sign in, whenever the user sign up, whenever the user sign out and whenever there is an authentication state change/ Authentication happens. And you what to do on this auth change if you want to handle that auth then use this onAuthStateChange API. This is kind of like an Event Listener
* So where we add this code?
* You can write wherever you want to but mostly use Root level to write this code so I am writing it on Body component

**#useNavigate Hook**

* <https://medium.com/@bobjunior542/using-usenavigate-in-react-router-6-a-complete-guide-46f51403f430>
  + **Sign-out** **a user call**[**signOut**](https://firebase.google.com/docs/reference/js/auth?authuser=0#signout)
    - import { getAuth, **signOut** } from "firebase/auth";
* **const handleSignOut = () => {**
* **signOut(auth)**
* **.then(() => {**
* **// Sign-out successful.**
* **navigate("/");**
* **})**
* **.catch((error) => {**
* **// An error happened.**
* **navigate("/error", error);**
* **});**
* **};**
  + **Update a user's profile**
  + You can update a user's basic profile information—the user's display name and profile photo URL—with the **updateProfile** method. For example:
* **// As Soon As new user Successfuly register then updating the profile with name and photoURL**
* **updateProfile(user, {**
* **displayName: name.current.value,**
* **photoURL: USER\_AVATAR,**
* **})**
* **.then(() => {**
* **// Profile updated!**
* **const { uid, email, displayName, photoURL } = auth.currentUser;**
* **dispatch(**
* **addUser({uid: uid, email: email, displayName: displayName, photoURL: photoURL})**
* **);**
* **})**
* **.catch((error) => {**
* **// An error occurred**
* **setErrorMessage(error.message);**
* **});**

**#Register TMDB API & create an app & get access token**

* TMDB login 🡺 Edit Profile 🡺 API 🡺 API key & API Read Access Token
* First you have to create app / register app then you will see all this details
* <https://www.themoviedb.org/settings/api>
* TMDB login 🡺 Edit Profile 🡺 API 🡺 Documentation 🡺 API Reference
* **The Movie Database (TMDB)** API. This is where you will find the definitive list of currently available methods for our movie, tv, actor and image API.
* <https://developer.themoviedb.org/reference/intro/getting-started>
* TMDB login 🡺 Edit Profile 🡺 API 🡺 Documentation 🡺 API Reference 🡺 MOVIE LISTS
  + Now Playing
  + Popular
  + Top Rated
  + Upcoming

#Custom Hook

* **Created Custom hook for each of Movie Lists**
  + Now Playing, Popular, Top Rated, Upcoming
    - This code is basically Fetching the data from TMDB and putting those movies into the store
* **#Introduction to GPT**
* We are going to build a separate page it is kind of like a Netflix GPT search page
* We are using OpenAI’s GPT
* We will have search bar and suppose sometimes we don’t know what or which movie to watch right! and I just want to type something into the search bar and want that GPT APIs should give me the results i.e. what movie to watch. It’s kind of like using GPT as movie recommendation
* For example: old Bollywood action movie

#GPT Search Feature

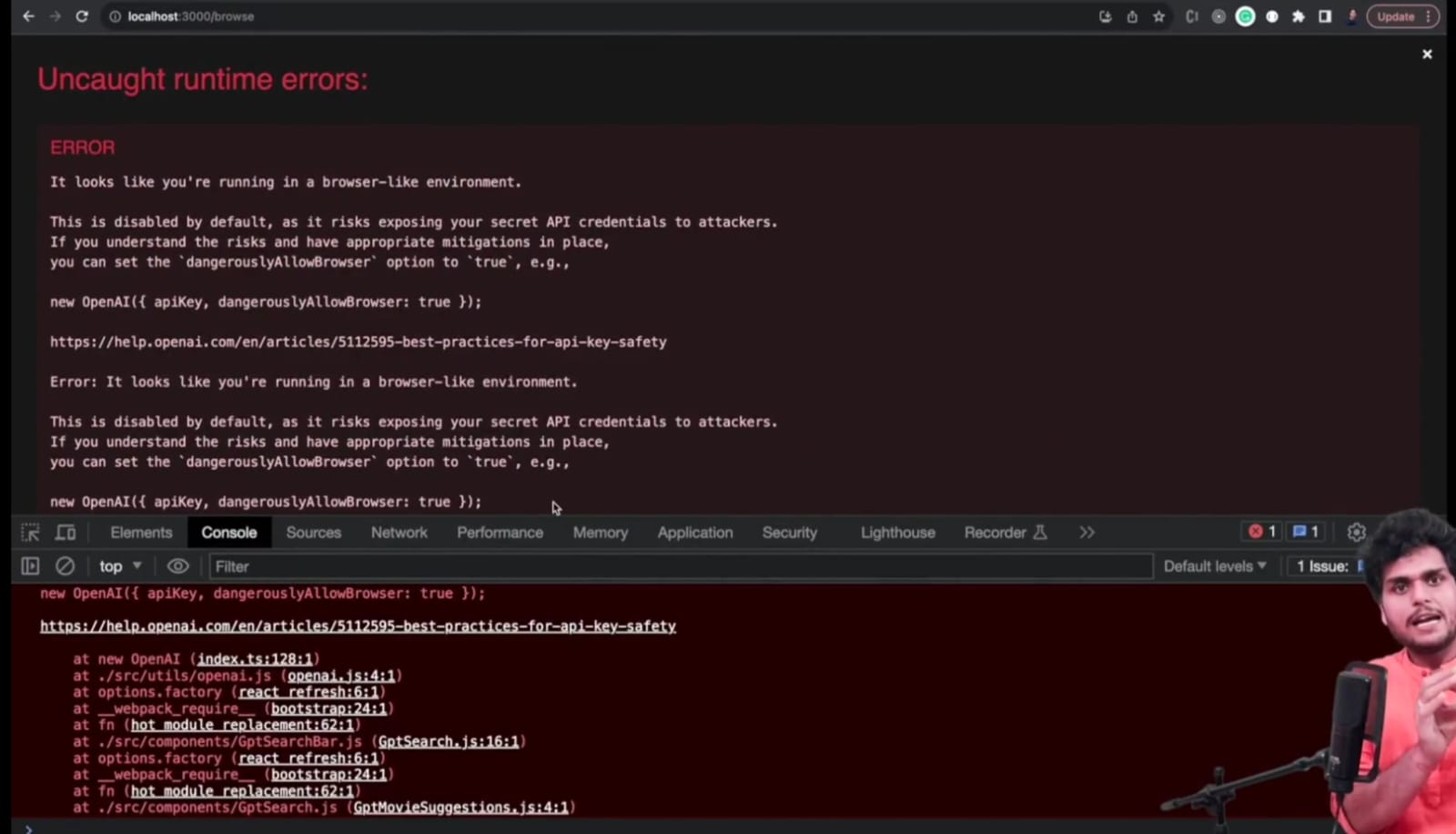
#Multi-language for GPT-Search-Page

**GPT API**

* [**https://platform.openai.com/**](https://platform.openai.com/) >> Login >> API Keys >>
* Create 🡺 NetflixGPTProject **API KEY** ==> sk-proj-TmTn0rsr28VWhSYErR20dAT3BlbkFJI0NzwPB0A4mcjxiw17XY
* But you have to pay for this API nearly 5$ ChatGPT API is not free
* We have to Install Library => **OpenAI Node API Library 🡺** <https://www.npmjs.com/package/openai>
  + **npm i --save openai**
* there is multiple model in GPT => GPT 3, GPT 3.5 Turbo, GPT 4
* we are using GPT 3.5 Turbo because GPT 4 is Expensive

**Error:** you will get this error and now understand why this error is there. OpenAI is trying to warn you that looks like you are calling this API.

* **const gptResults = await openai.chat.completions.create({**
* **messages: [{ role: "user", content: gptQuery }],**
* **model: "gpt-3.5-turbo",**
* **});**
* So basically our react app runs from the front-end on the browser and I am calling this API from a browser like Environment
* So understand the difference I am calling it from the client side not from the server side, it is basically saying that my API Secret key can be leaked because this API key is in a react app
* But don’t you worry also tells you that if you want to allow this then you can set this **dangerously allow browser true**



* In openai.js file [OpenAI Configuration]
* **import OpenAI from "openai";**
* **import { OPENAI\_KEY } from "./constant";**
* **//it is kind of like Authorization function - Helper function**
* **const openai = new OpenAI({**
* **apiKey: OPENAI\_KEY,**
* **dangerouslyAllowBrowser: true,**
* **});**
* **export default openai;**

#Good-Prompt – GptQuerys

* **const gptQuery =**
* **"Act as Movie Recommendation System and Suggest some Movies for the Query : " +**
* **searchText.current.value +**
* **". only give me names of 5 movies, comma seperated like the example result given ahead. Example Result: Gadar, Sholay, Don, Golmaal, Koi Mil Gaya";**

GPT API

# Search Movie

* TMDB login 🡺 Edit Profile 🡺 API 🡺 Documentation 🡺 API Reference 🡺 Search 🡺 Movie
* **// Searching Movie in this TMDB-API**
* **const searchMovieTMDB = async (movie) => {**
* **const data = await fetch("https://api.themoviedb.org/3/search/movie?query="+movie+"&include\_adult=false&language=en-US&page=1", API\_OPTIONS );**
* **const json = await data.json();**
* **return json.results;**
* **};**

**#SearchBar Component**

**import React, { useRef } from "react";**

**import lang from "../../utils/languageConstants";**

**import { useDispatch, useSelector } from "react-redux";**

**import openai from "../../utils/openai";**

**import { API\_OPTIONS } from "../../utils/constant";**

**import { addGptMovieResult } from "../../utils/gptSlice";**

**const GptSearchBar = () => {**

**const dispatch = useDispatch();**

**const langKey = useSelector((store) => store.config.lang);**

**const searchText = useRef(null);**

**// Searching Movie in this TMDB-API**

**const searchMovieTMDB = async (movie) => {**

**const data = await fetch(**

**"https://api.themoviedb.org/3/search/movie?query="+movie+"&include\_adult=false&language=en-US&page=1",**

**API\_OPTIONS);**

**const json = await data.json();**

**// console.log("json.results:", json);**

**return json.results;**

**// console.log("first", json);**

**};**

**searchMovieTMDB();**

**const handleGptSearchClick = async () => {**

**console.log(searchText.current.value);**

**const gptQuery =**

**"Act as Movie Recommendation System and Suggest some Movies for the Query : " +**

**searchText.current.value +**

**". only give me names of 5 movies, comma separated like the example result given ahead. Example Result: Gadar, Sholay, Don, Golmaal, Koi Mil Gaya";**

**// make an API call to GPT-API and get movie results**

**const gptResults = await openai.chat.completions.create({**

**messages: [{ role: "user", content: gptQuery }],**

**model: "gpt-3.5-turbo",**

**});**

**if (!gptResults.choices) {**

**// Handle Error**

**console.log("The Movies are not available!!!!!!!!!!!!!!!!!!!!!!!!");**

**}**

**const gptMovies = gptResults.choices?.[0]?.message?.content.split(",");**

**console.log("gptResults.choices GetMovies:", gptMovies);**

**// console.log("gptResults.choices GetMovies:", gptMovies);**

**// for each movie we will search TMDB-API**

**const promiseArray = gptMovies.map((movie) => searchMovieTMDB(movie));**

**// here in "promiseArray" we are getting  5 promise i.e. for 5 movies**

**//Promise.all() it takes array of promises it will only finish once all these 5 promises are resolved**

**const tmdbResults = await Promise.all(promiseArray);**

**console.log("tmdbResults:", tmdbResults);**

**dispatch(**

**addGptMovieResult({ movieNames: gptMovies, movieResults: tmdbResults })**

**);**

**};**

Q: promise and promiseAll ??

.env file

* Whatever secret key you have you have to keep it in env file
  + You have to save file like this you have to add REACT\_APP in front of it.
    - REACT\_APP\_KEYNAME=key
  + And wherever you are using use it like this
    - process.env.REACT\_APP\_KEYNAME

#Memoization

* Memoization is a very heavy word but it is very easy
* Whenever we go-to our home page it keep making API calls then why it making an API call again and again?
* See it making an API call because every time my component loads the hook is called and my store is updated but-but-but if we see clearly when we again go back to home the data is already there in store, if the data is present why to make unnecessary API calls again and again
* So we can use the concept of Memoization to stop this. How to do that?
* Suppose my store already had that trailer video why to fetch it again?
* I will check whether that trailer-video are there in my store or not? If they are already present in my store then I will not make an API call right!!

**const trailerVideo = useSelector((store) => store.movies.trailerVideo);**

* If the trailer-video is not empty if it has some value so don’t make an API call. We can write this logic
* If trailerVideo is not there then only make an API Call

**useEffect(() => {**

**// call only when trailerVideo is not present(memoization)**

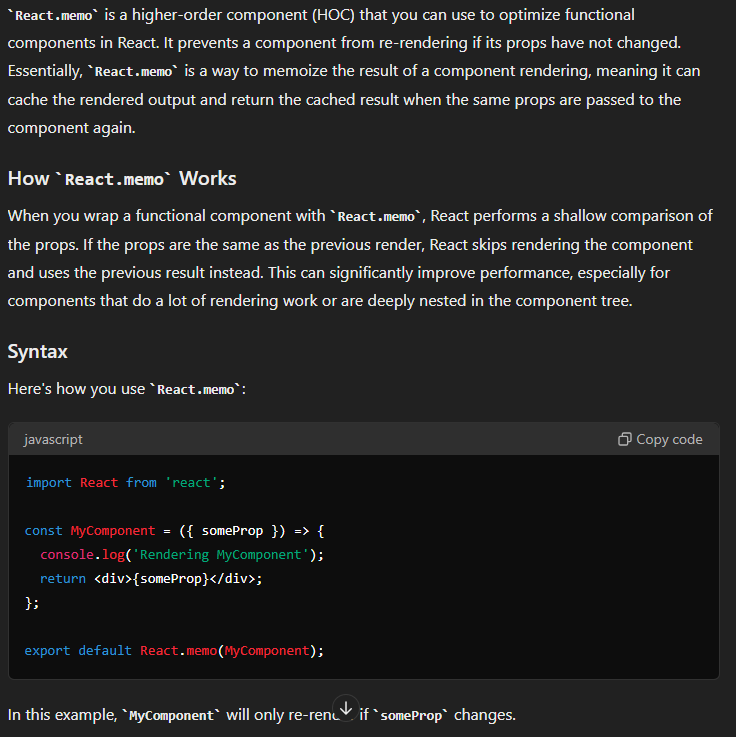
**// if (!trailerVideo) getMovieVideo();**

**!trailerVideo && getMovieVideo();**

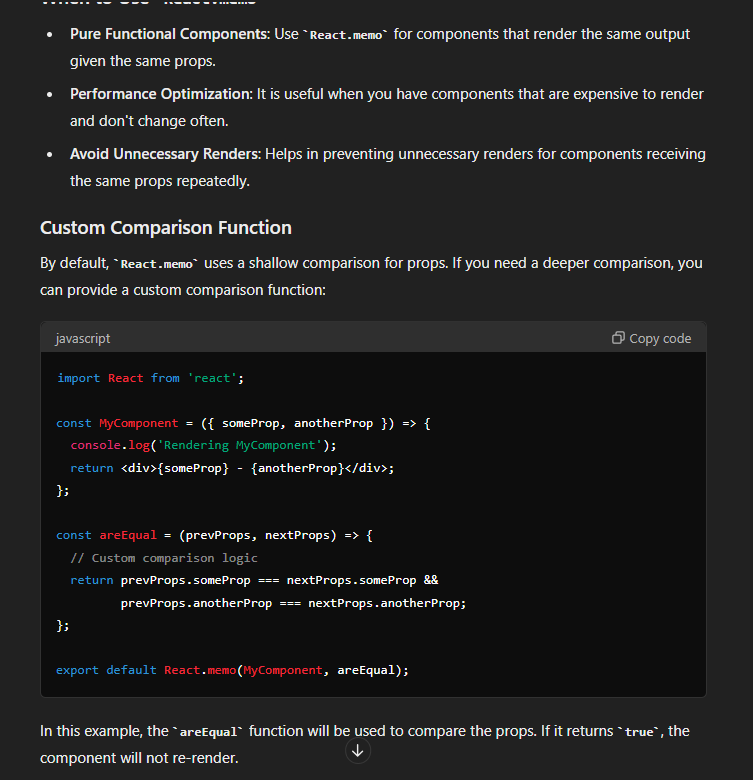
**}, []);**

* Its only be called if trailer-video does not have data

**React.memo()**



**# When to use React.memo**



**Limitations**

* **State Changes**: **React.memo** only optimizes props; it doesn't help with state changes within the component.
* **Deep Prop Changes**: For deeply nested objects, the default shallow comparison might not be sufficient, and you might need to provide a custom comparison function.
* **Overhead**: There is some overhead associated with **React.memo** due to the comparison logic. It should be used when you have identified performance bottlenecks.



### Conclusion

**React.memo** is a powerful tool for optimizing React functional components by preventing unnecessary re-renders. It's especially useful for components that render based on props and don't have internal state changes. By using React.memo, you can improve the performance of your React applications, especially when dealing with complex component trees or expensive rendering logic.