**Start 17:45**

**Redux**

Advantage

 **Predictable State Management:** Redux enforces a strict unidirectional data flow, which can make your application more predictable and easier to debug.

 **Middleware:** Redux has a powerful middleware ecosystem (e.g., redux-thunk, redux-saga) that allows for handling asynchronous logic and side effects.

 **DevTools:** Redux DevTools provide a robust set of tools for debugging and inspecting the state changes in your application.

 **Community and Ecosystem:** Redux has a large community, extensive documentation, and a wealth of third-party libraries and resources.

Disadvantage

 **Boilerplate Code:** Redux often requires a significant amount of boilerplate code for setting up actions, reducers, and stores.

 **Complexity:** For simpler applications, Redux can be overkill and unnecessarily complex.

**Zustand**

Advantage

 **Simplicity:** Zustand is known for its simplicity and minimal boilerplate. It provides a straightforward API for managing state.

 **Performance:** Zustand is highly performant, making it suitable for applications that require fast state updates.

 **Size:** Zustand is lightweight compared to Redux, which can be beneficial for reducing the overall bundle size of your application.

 **No Boilerplate:** Zustand requires minimal setup and does not impose a strict structure, allowing for more flexibility in how you manage state.

Disadvantage

 **Less Opinionated:** While flexibility is a pro, it can also be a con if you prefer the structured approach that Redux enforces.

 **Smaller Ecosystem:** Zustand has a smaller community and fewer third-party integrations compared to Redux.

 **Less Mature:** While Zustand is gaining popularity, it doesn't have the same level of maturity and widespread adoption as Redux.

**Recoil**

 **Atoms and Selectors:** Recoil introduces the concepts of atoms and selectors. Atoms are units of state that can be shared across components, while selectors are pure functions that can derive state from atoms and other selectors.

 **Reactivity:** When an atom's state changes, all components that subscribe to that atom are automatically re-rendered, providing a reactive state management system.

 **Simplicity:** Recoil is designed to be simple and intuitive, reducing the boilerplate code needed to manage state.

 **Concurrency:** Recoil supports concurrent mode in React, making it compatible with future React features.

 **Performance:** Recoil is optimized for performance, ensuring efficient updates and minimal re-renders.

 **Data Flow Graph:** Recoil builds a data flow graph where atoms and selectors are nodes, allowing for more efficient and granular updates.

Advantage

 **Easy to Use:** Recoil's API is straightforward and easy to learn, making it accessible for developers of all experience levels.

 **Flexible:** Recoil offers flexibility in how you structure your state, without enforcing a rigid architecture.

 **Efficient Re-renders:** The reactivity model minimizes unnecessary re-renders, improving performance.

 **Integration with Concurrent Mode:** Recoil is built to work seamlessly with React's concurrent mode, preparing your app for future React advancements.

disadvantage

 **Smaller Ecosystem:** Compared to Redux, Recoil has a smaller ecosystem and community, which might mean fewer resources and third-party libraries.

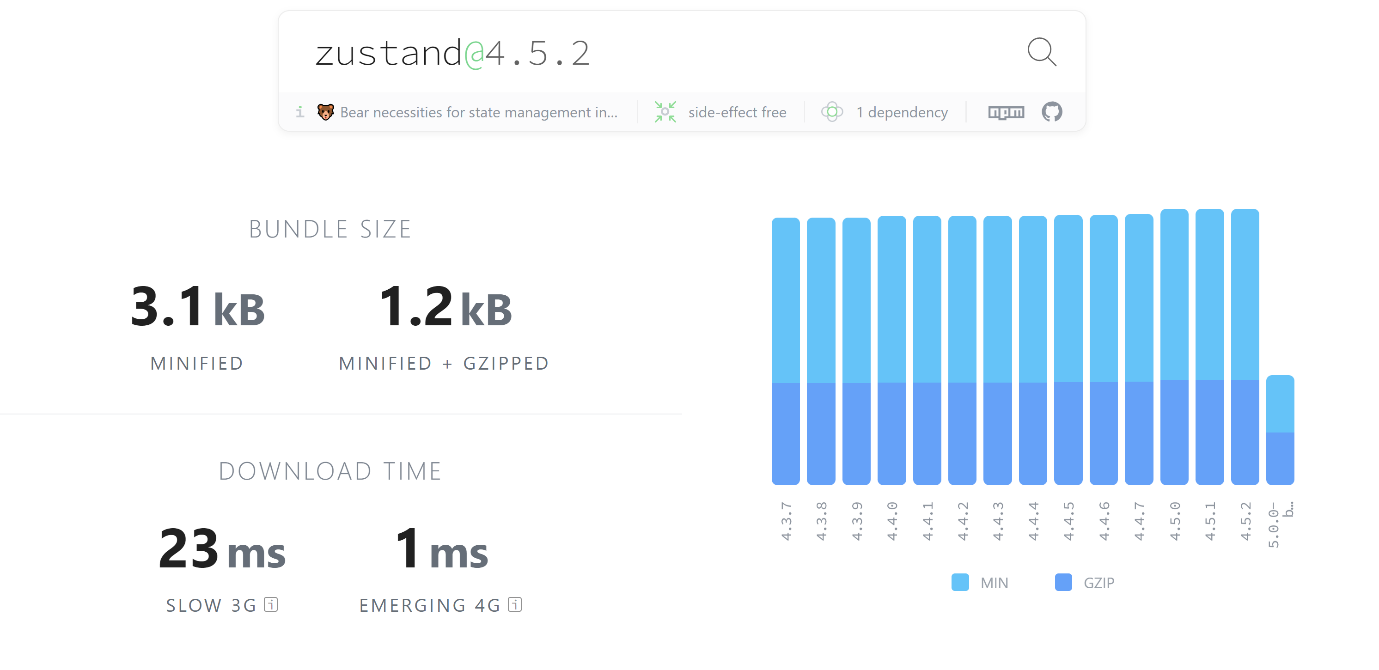
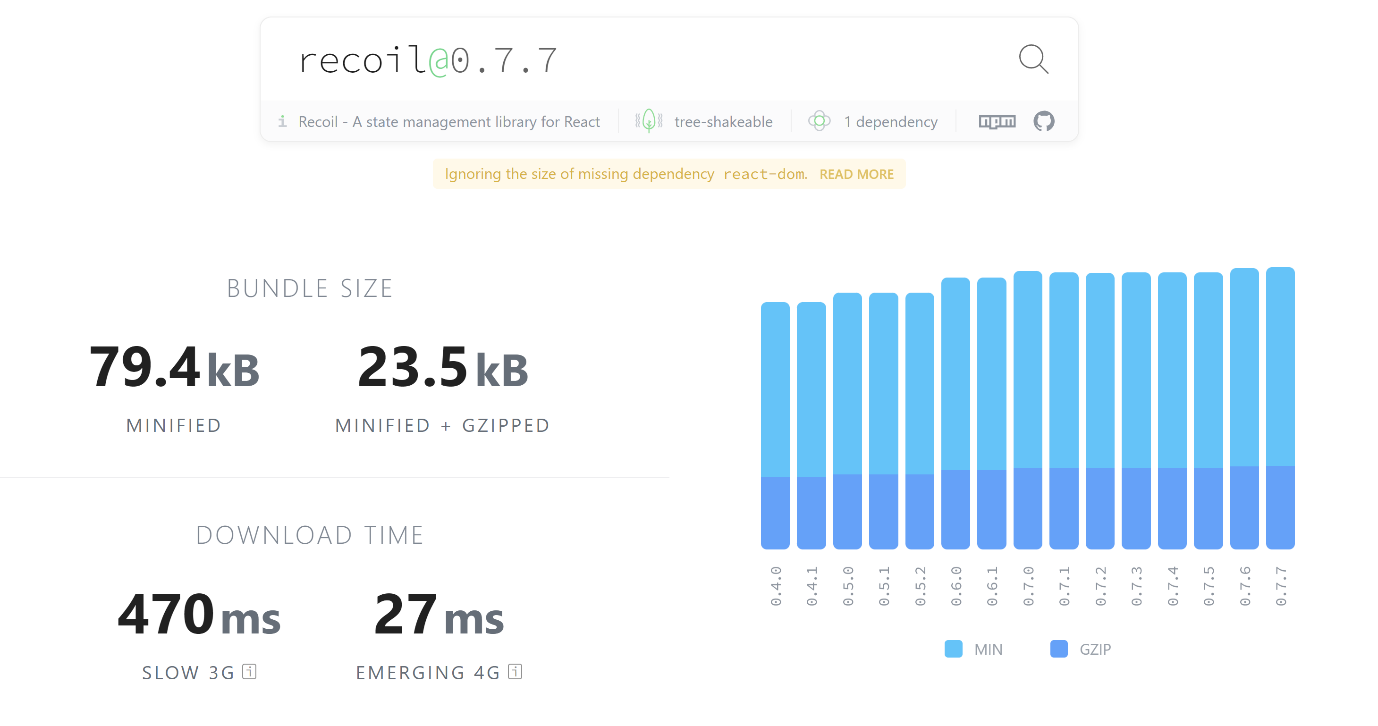
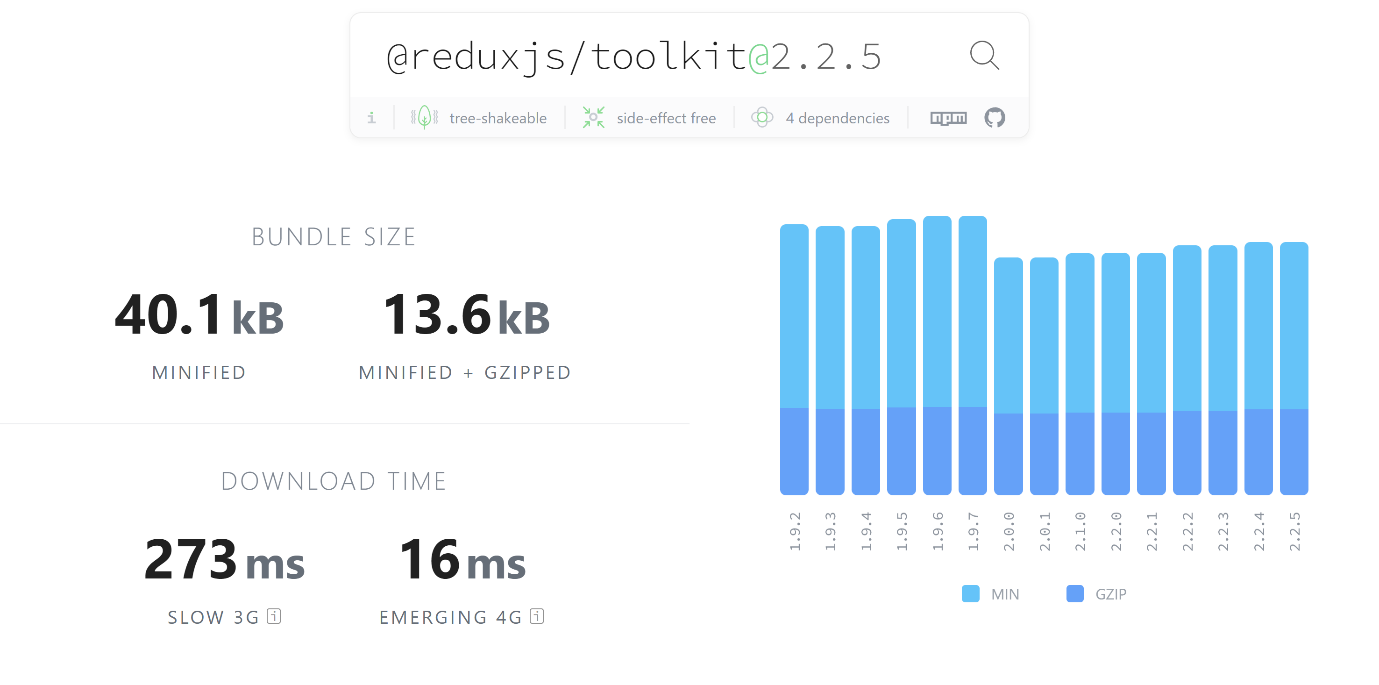
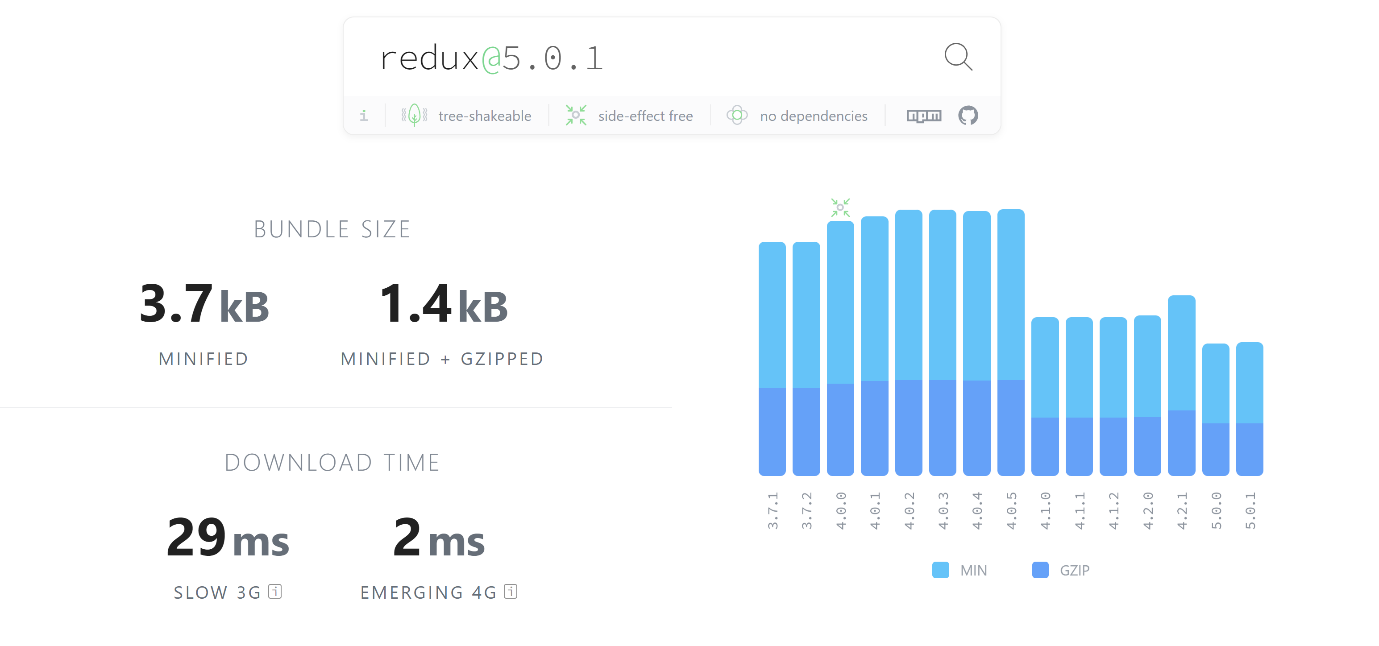
 **Less Mature:** Recoil is relatively new compared to Redux, so it might lack some of the battle-tested stability and extensive documentation of older libraries.

 **Facebook-Specific Use Cases:** As it was developed by Facebook, some features or design decisions might be optimized for Facebook’s specific use cases, which may not always align with your needs.

Npm trends

[**https://npmtrends.com/@reduxjs/toolkit-vs-recoil-vs-redux-vs-zustand**](https://npmtrends.com/@reduxjs/toolkit-vs-recoil-vs-redux-vs-zustand)

**BUNDLE SIZE**



Extention

Import Cost

Resources

# [Zustand Beginner Tutorial - Learn React State Management With Zustand](https://www.youtube.com/watch?v=-Y8brhQKvtA)

Npm

<https://www.npmjs.com/package/zustand>

npm install zustand

import { create } from 'zustand'

const useBearStore = create((set) => ({

bears: 0,

increasePopulation: () => set((state) => ({ bears: state.bears + 1 })),

removeAllBears: () => set({ bears: 0 }),

}))

npm i immer

<https://docs.pmnd.rs/zustand/guides/auto-generating-selectors#vanilla-store>

8:10