in/poyrazavsever

STATE MANAGEMENT KARŞILAŞTIRMASI

Redux • Zustand • Jotai • Signals

in/poyrazavsever

Neden State Management?

- UI = f(state). Artan modül sayısı, karmaşık veri akışları.
- Paylaşılan/yerel state ayrımı kritik.
- Amaç: tahmin edilebilirlik, performans, bakım kolaylığı.

Seçimi etkileyenler: Ekip deneyimi, uygulama boyutu, veri karmaşıklığı, SSR/CSR, platform.

in/poyrazavsever

Redux

Artılar: Tek yönlü akış, güçlü tooling, ekosistem zengin.

Eksiler: Boilerplate, action/reducer kavram yükü.

Ne Zaman? Kurumsal projeler, karmaşık akışlar, büyük ekipler

```ts
// store.ts
import { configureStore, createSlice } from '@reduxjs/toolkit'
const counter = createSlice({
  name: 'counter',
  initialState: { value: 0 },
  reducers: { inc: s => { s.value++ } }
})
export const { inc } = counter.actions
export const store = configureStore({ reducer: { counter: counter.reducer } })
```

in/poyrazavsever

Zustand

Artılar: Minimal API, boilerplate yok, selector ile ince taneli render.

Eksiler: Büyük ekiplerde kuralları siz koyarsınız.

Ne Zaman? Orta ölçek, hızlı prototip, mikro-frontend modülleri.

```
// store.ts
import { create } from 'zustand'
export const useCounter = create((set)=>({
  value: 0,
  inc: () => set(s => ({ value: s.value + 1 }))
}))
```

in/poyrazavsever

Jotai

Artılar: Primitive-first; state = atom. Kompozisyon kolay, React concurrent uyumu iyi.

Eksiler: Mimaride sınırlar ekipçe tanımlanmalı.

Ne Zaman? Bileşen odaklı, bağımsız parçalarda.

```tsx
// counter.tsx
import { atom, useAtom } from 'jotai'
const countAtom = atom(0)
export default function Counter(){
  const [count, set] = useAtom(countAtom)
  return <button onClick={()=>set(c=>c+1)}>Count: {count}</button>
}
```

in/poyrazavsever

Signals

Fikir: Reaktif primitive; bağımlılık izleme ile hassas güncelleme.
Artılar: Çok ince taneli re-render, yüksek performans.
Eksiler: Ekosistem/şablon farklılıkları, öğrenme eğrisi.
Ne Zaman? Yoğun etkileşim, yüksek frekanslı UI güncellemeleri.

Performans Karşılaştırma

- **Redux:** Memoization ve selector’larla iyi; global güncellemelerde geniş yayılım.
- **Zustand:** Selector tabanlı abonelik, daha az render.
- **Jotai:** Atom bazlı ince taneli güncelleme.
- **Signals:** Pull yerine push-like reaktivite; minimum kirlenme.

Ölçeklenebilirlik & Ekip

- **Redux:** Sıkı kurallar, kolay code review, büyük ekip uyumu.
- **Zustand/Jotai:** Esneklik yüksek; style-guide şart.
- **Signals:** Reaktif mimari bilgisi önemli.

Peki Siz

hangi yaklaşımı kullanıyorsunuz ve

neden?

in/poyrazavsever