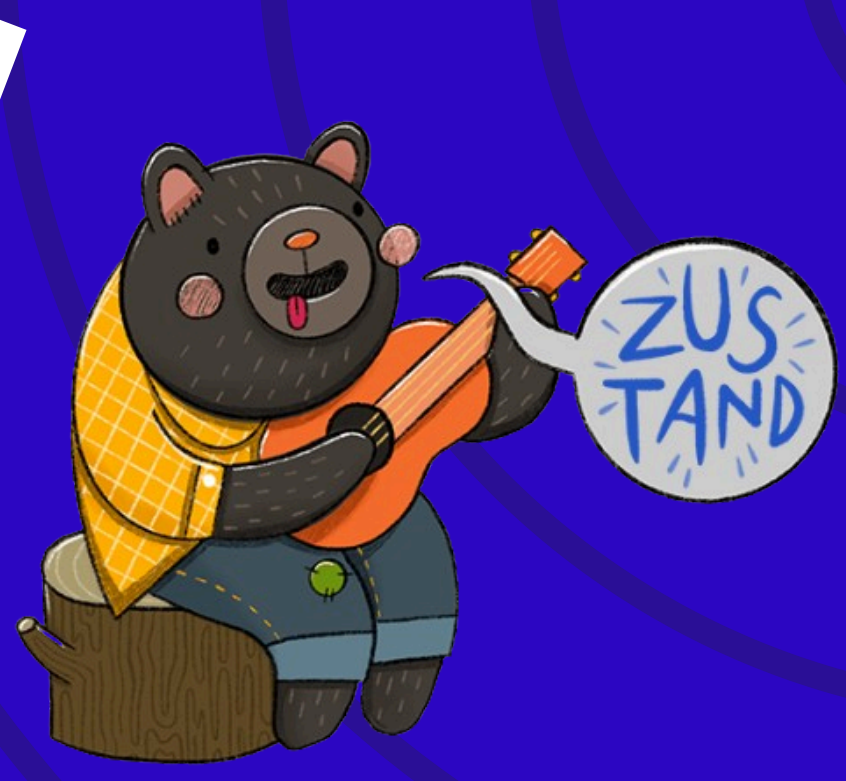
FULL COMPARISON

REDUX VS ZUSTAND

STATE MANAGEMENT IN REACT



SAMMAN

CREATOR TIPS

FULL COMPARISON

REDUX

is a predictable state container for JavaScript apps. It helps you manage and centralize the application state using actions, reducers, and a global store.

KEY REDUX CONCEPTS:

- **Centralized State:** Stores the entire app state in a single, immutable store for consistency.
- **Actions & Reducers**: State changes happen through dispatched actions handled by pure reducers.
- **Boilerplate**: Requires more setup with separate files for actions, reducers, and store config.

FULL COMPARISON

ZUSTAND

is a lightweight, hook-based state management library that simplifies global state without context, actions, or reducers. It's fast, minimal, and easy to use for any React project.

KEY REDUX CONCEPTS:

- **Minimal Setup:** Create a store with a single create() function.
- **Direct State Manipulation**: State is mutated directly in the store, without actions or reducers.
- **No Boilerplate**: Simple, intuitive API with minimal setup.

FULL COMPARISON

FEATURES COMPARISON

Aspect	REDUX	ZUSTAND
Learning Curve	Steep for newbies	Very easy
Boilerplate	Heavy setup	Easy setup
Small App Suitability	Overkill	Ideal
Middleware/ Async Handling	Built-in (e.g., Thunk, Saga)	Manual setup needed
Handling Complex State	Great for large apps	Not ideal for deeply complex logic

SETUP

REDUX :

Actions.js

```
export const increment = ()=>({ type: 'INCREMENT'});
export const decrement = ()=>({ type: 'DECREMENT'});
```

Reducer.js

```
const counterReducer = (state = 0, action)=>{
switch(action.type){
case'INCREMENT':
return state + 1;
case'DECREMENT':
return state - 1;
default:
return state;
}
};
export default counterReducer;
```

ZUSTAND:

- Zustand is just a hook — no provider, no reducers, no actions.

STORE SETUP

REDUX:

- Store.js

```
 import{ configureStore } from '@reduxjs/toolkit';
import counterReducer from './reducer';
// Create Redux store using configureStore
const store = configureStore({
  reducer: counterReducer
});
export default store;
```

ZUSTAND:

- Store.js

```
import{ create } from 'zustand';
const useStore = create((set)=>({
  count: 0,
  increment: ()=>set((state)=>({ count: state.count + 1})),
  decrement: ()=>set((state)=>({ count: state.count - 1}))
}));
export default useStore;
```

Async Logic Handling

REDUX:

- Use createAsyncThunk() or redux-saga / redux-thunk

```
export const fetchUser = createAsyncThunk(
  'user/fetchUser',
  async (id) => {
    const res = await fetch(`/api/user/${id}`);
    return res.json();
  }
);
```

ZUSTAND:

- use async functions directly inside the store.
- No middleware required.

```
const useStore = create(set => ({
  user: null,
  fetchUser: async (id) => {
    const res = await fetch(`/api/user/${id}`);
    const data = await res.json();
    set({ user: data });
  }
}));
```

When to Use What?

Use REDUX

- is great for structured, large-scale apps
- offers a ton of features, making it perfect for big, complex applications, but it requires time and effort to learn.

Use ZUTAND

- shines in simplicity, speed, and flexibility
- is the go-to for quick, simple projects, giving you fast setup and ease of use with minimal fuss.

SAMMAN

CREATOR TIPS

LINKEDIN

FOLLOW TO GET MORE INFORMATION AND TIPS LIKE THESE.

SAVE & SHARE