A quick guide about **Redux** &

Redux Toolkit(RTK)

In React / TypeScript

WHAT IS REDUX?

1. **Predictable State Management** by maintaining a single source of truth, it becomes easier to understand and debug how data changes over time. Redux's immutability and **unidirectional data flow** ensure **consistent** and **reliable state management**.

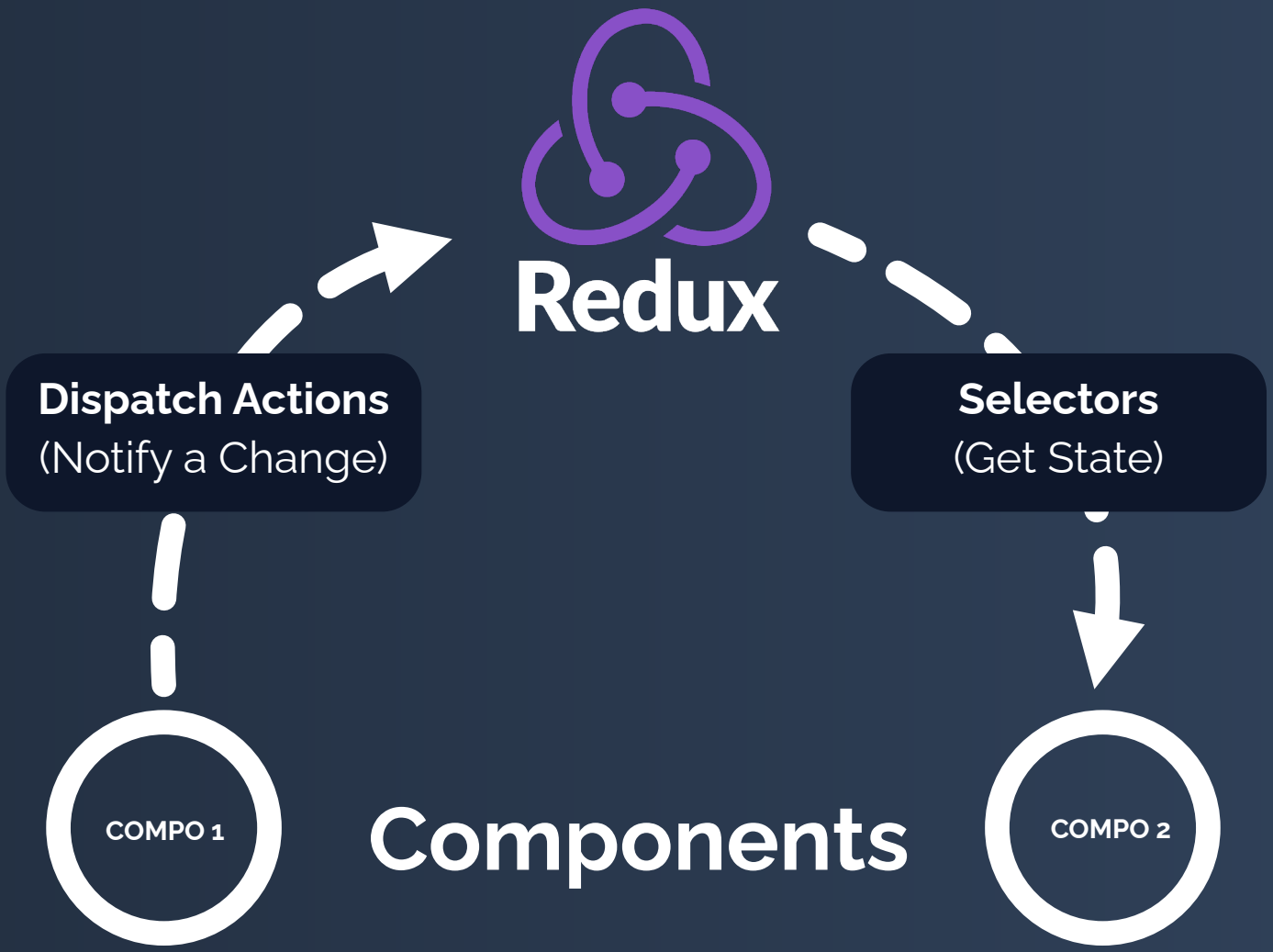
2. **Centralized Store:** this eliminates the need to pass data through multiple components, simplifying data flow and making it easier to access and update the state from anywhere in your app.

3. **Efficient Updates with Reducers:** Redux utilizes reducers to handle state updates. These **pure functions** provide a clear and structured way to modify the state, making it easier to maintain and reason about changes in your app

4. **Time Travel Debugging:** Redux's **powerful DevTools** extension allows for time travel debugging. You can **replay actions** and **inspect** the **state** at any point in time, making it a breeze to track down bugs and understand how your application's state evolves

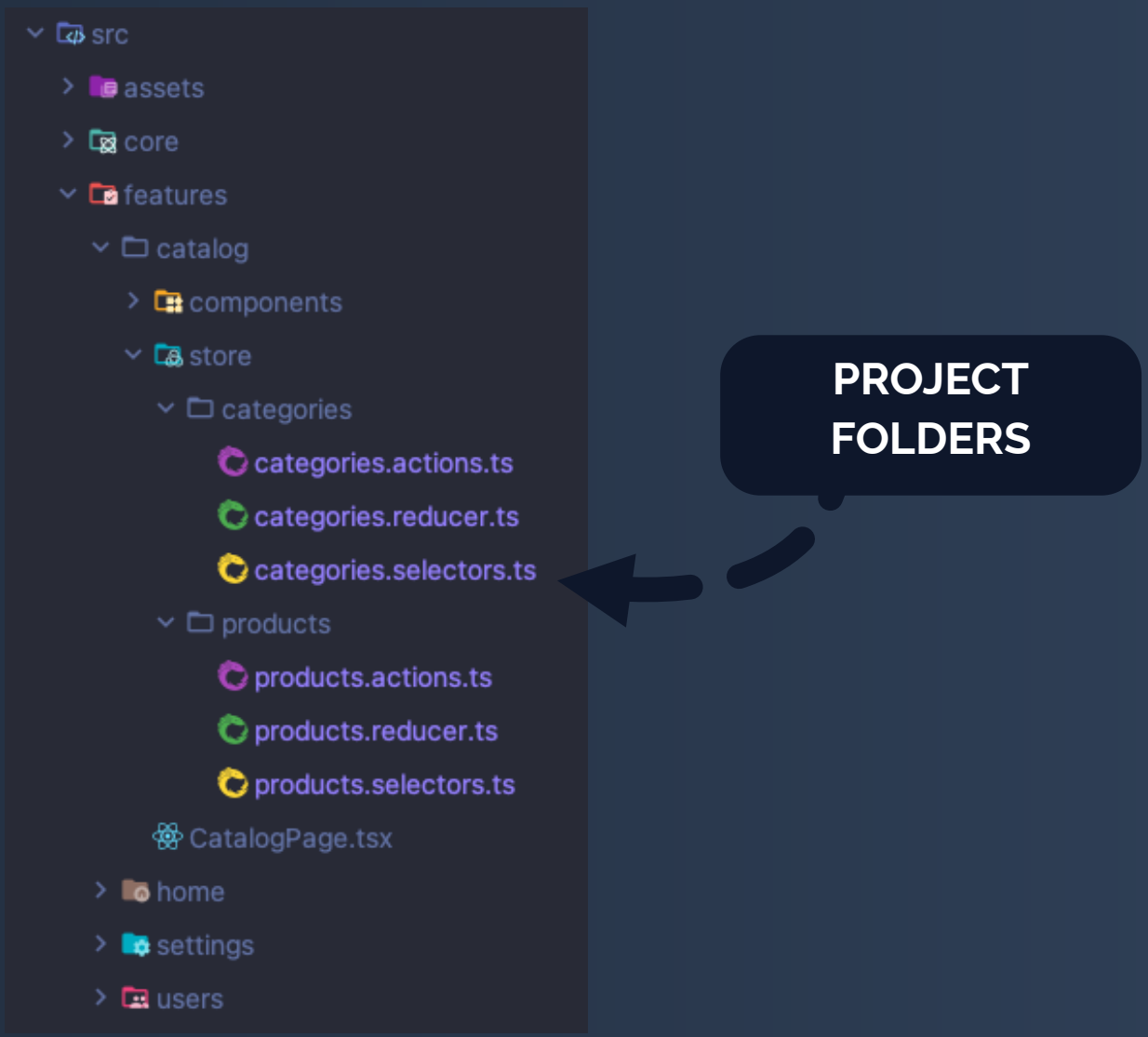
REDUX PRINCIPLES

- **Single Source of Truth**
  (one global state aka STORE)
- **State is read-only**
  (only actions can update the state)
- **Changes are made with pure functions**
  (using reducers)



WHY REDUX?

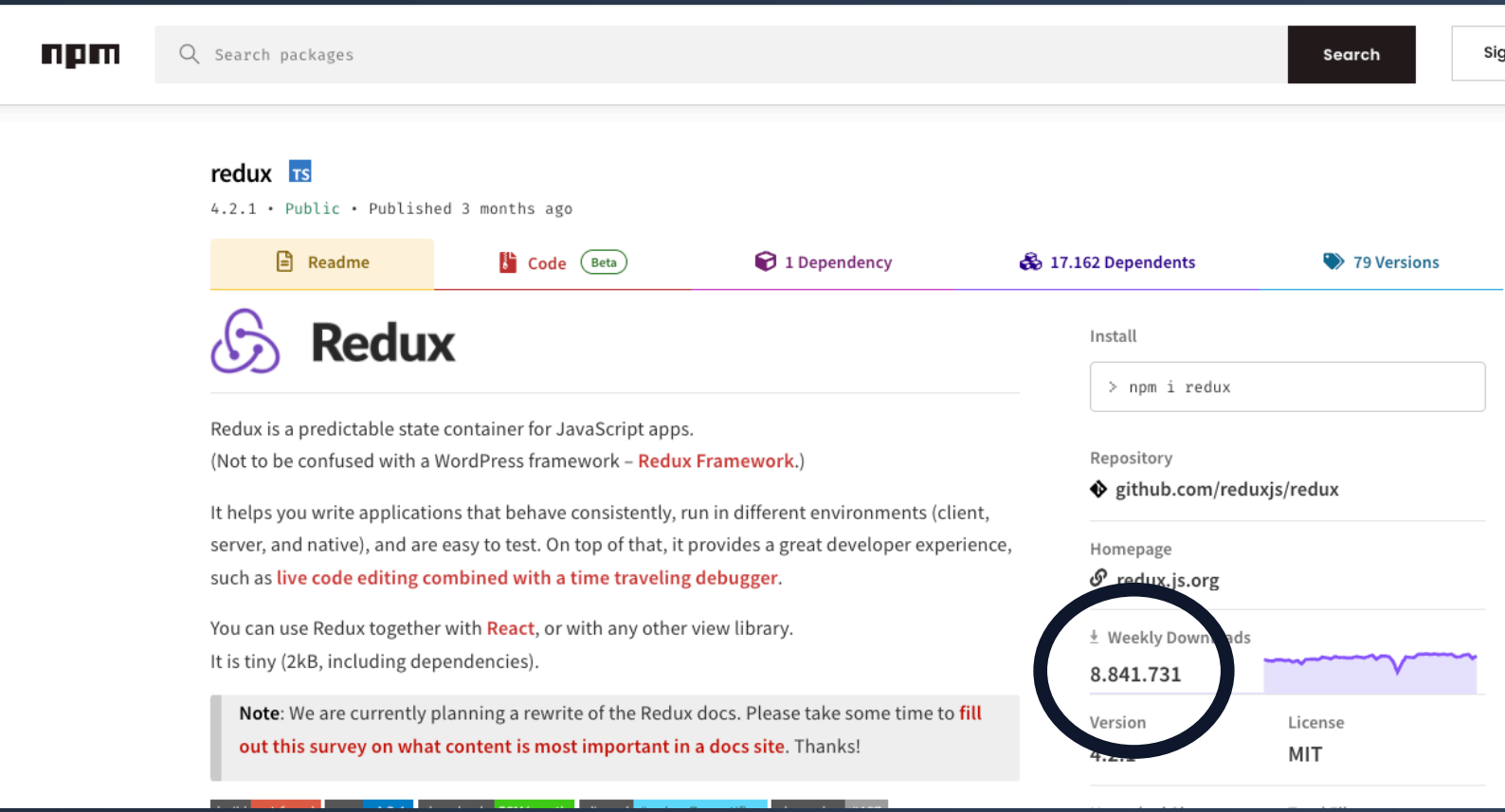
- State is **centralized** and **global**
- It's a well known **pattern**
- **Opinionated**: it helps developers to have some discipline
- **Don't reinvent the wheel**: for most commod scenarios can easily understand the architecture. Why?
- **Redux Dev Tools** (AMAZING!! See next slides)
- **Avoid unnecessary renders**



Who uses Redux?

Facebook, Netflix, AirBnb, Uber, Pinterest, Microsoft, GotoMeeting, Atlassian, StackBlitz are some of the companies that are using Redux.

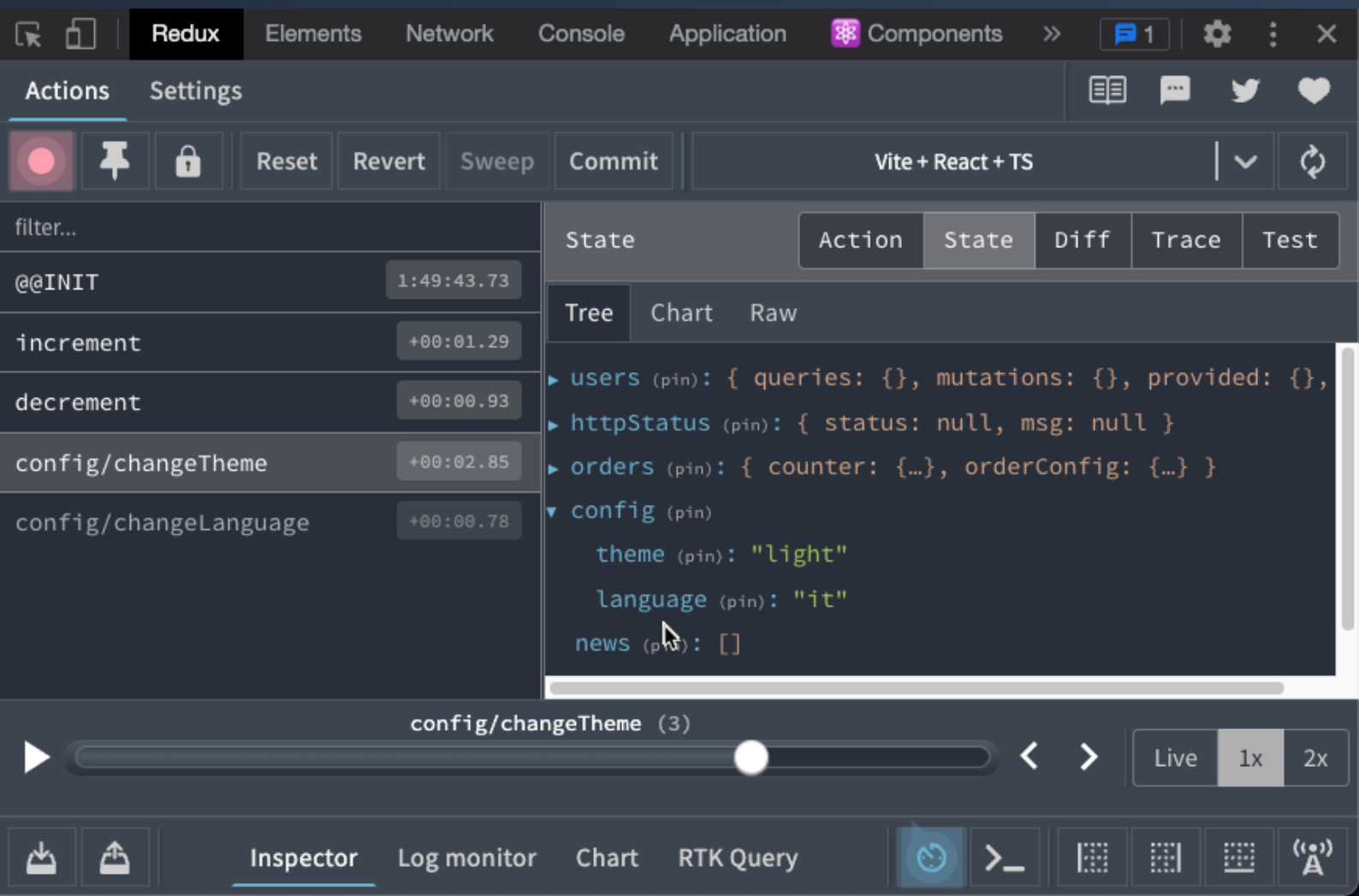
8.8M dowload / week



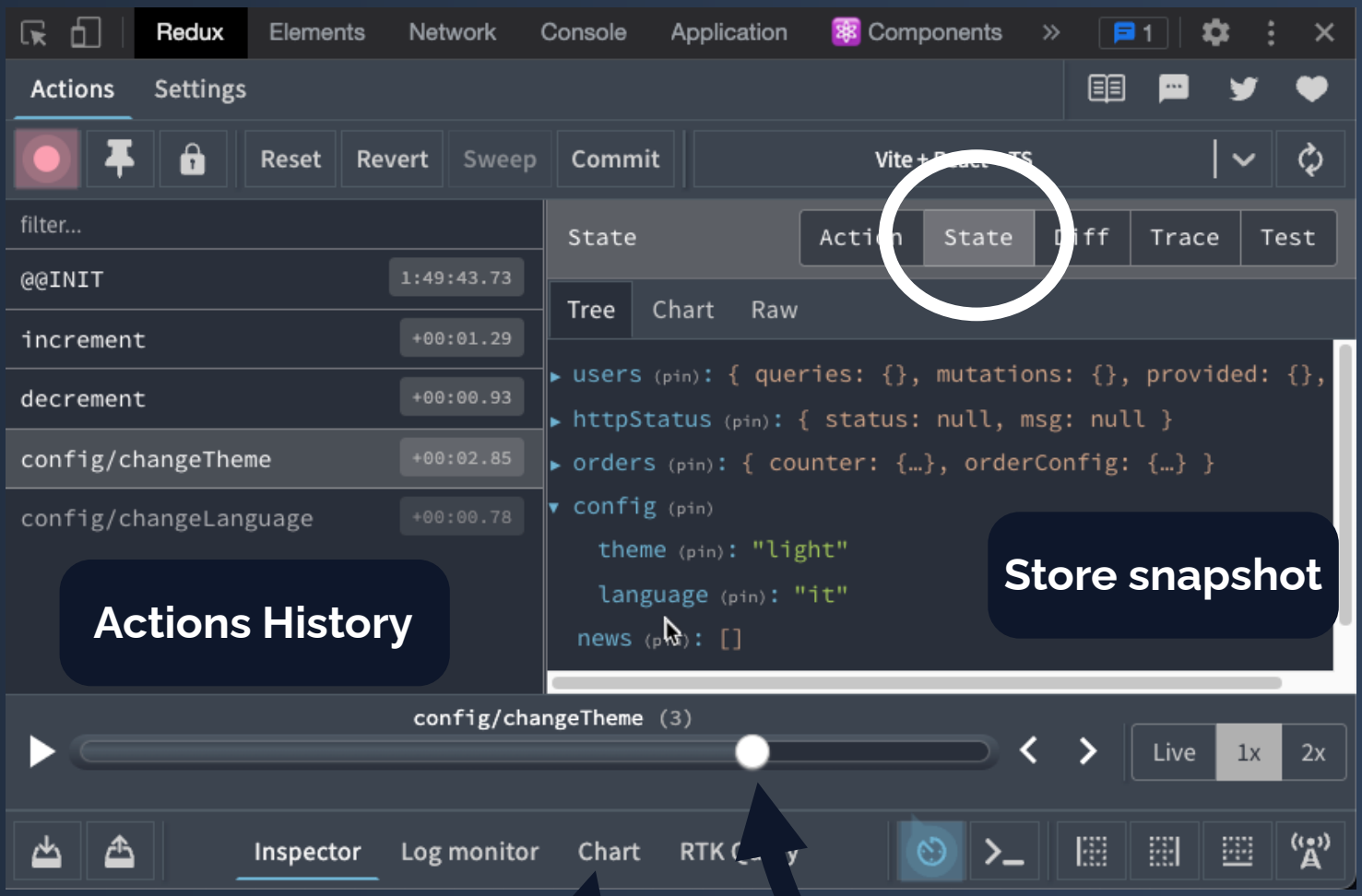
Redux DevTools

One of the main benefits using Redux is **Redux DevTools**, a browser extension that allow you to:

- Easily **debug** your Redux app
- Always know what is happening in your app
- **Track actions** and how the **state changes**
- **time travel** debugging
- Export a **snapshot** and load it later to re-create a scenario
- display the state **chart**
- and more...

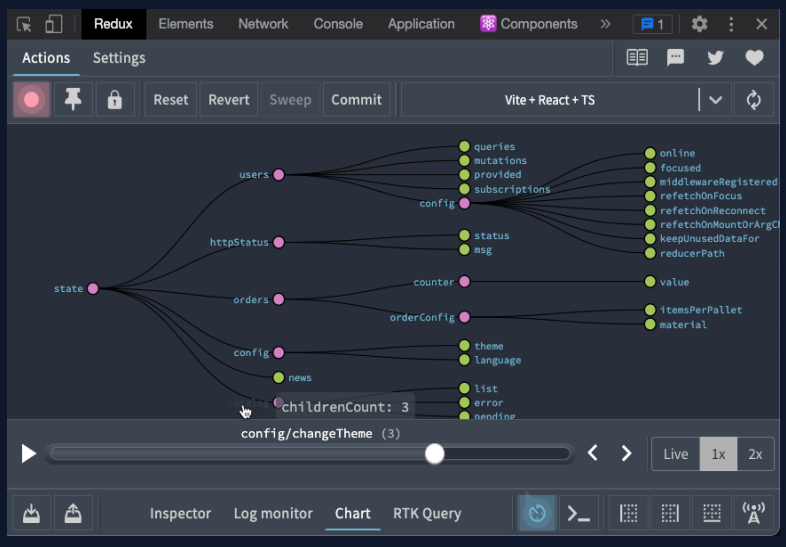


You can track everything that happens in your app



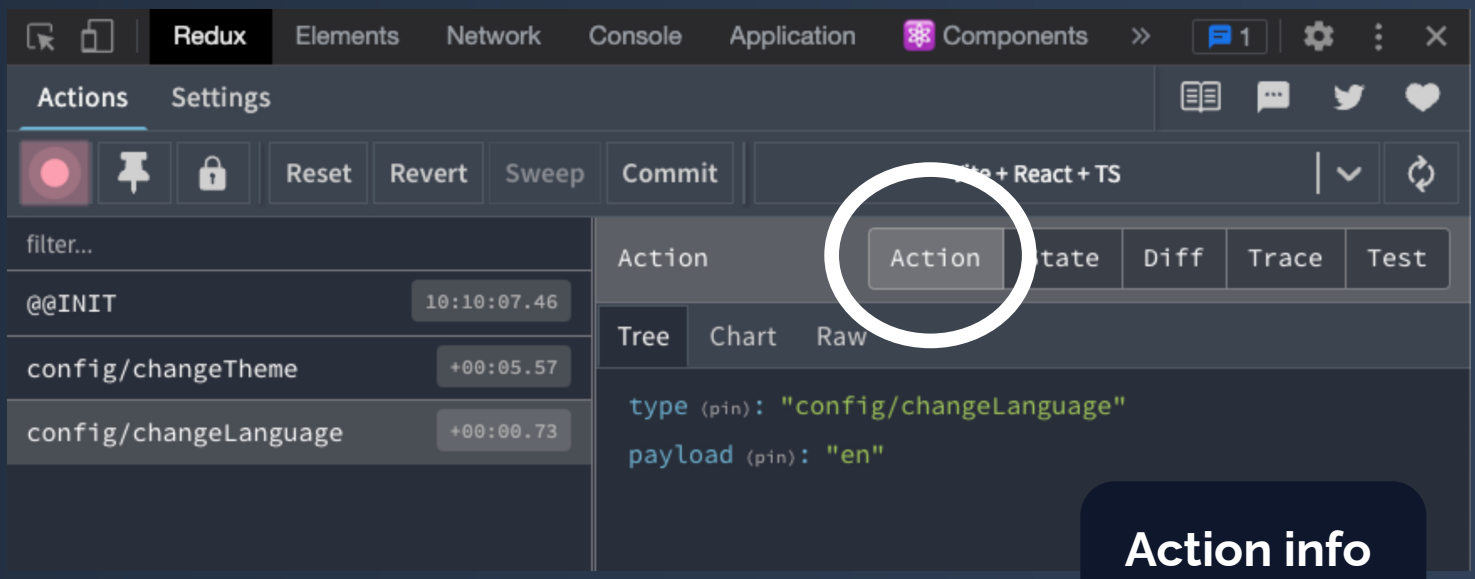
Time Travel Debug

State Diagram

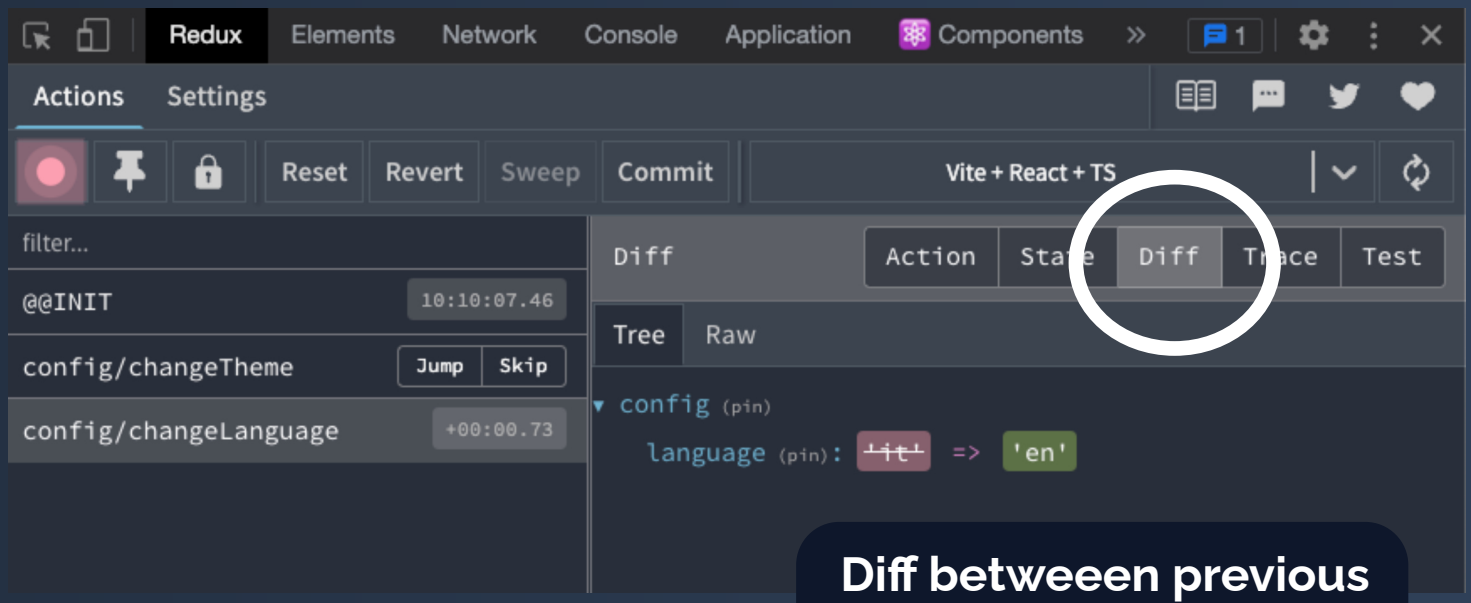


Redux DevTools

Track what actions you're dispatching (**type**)
and which parameters are you sending (**payload**)

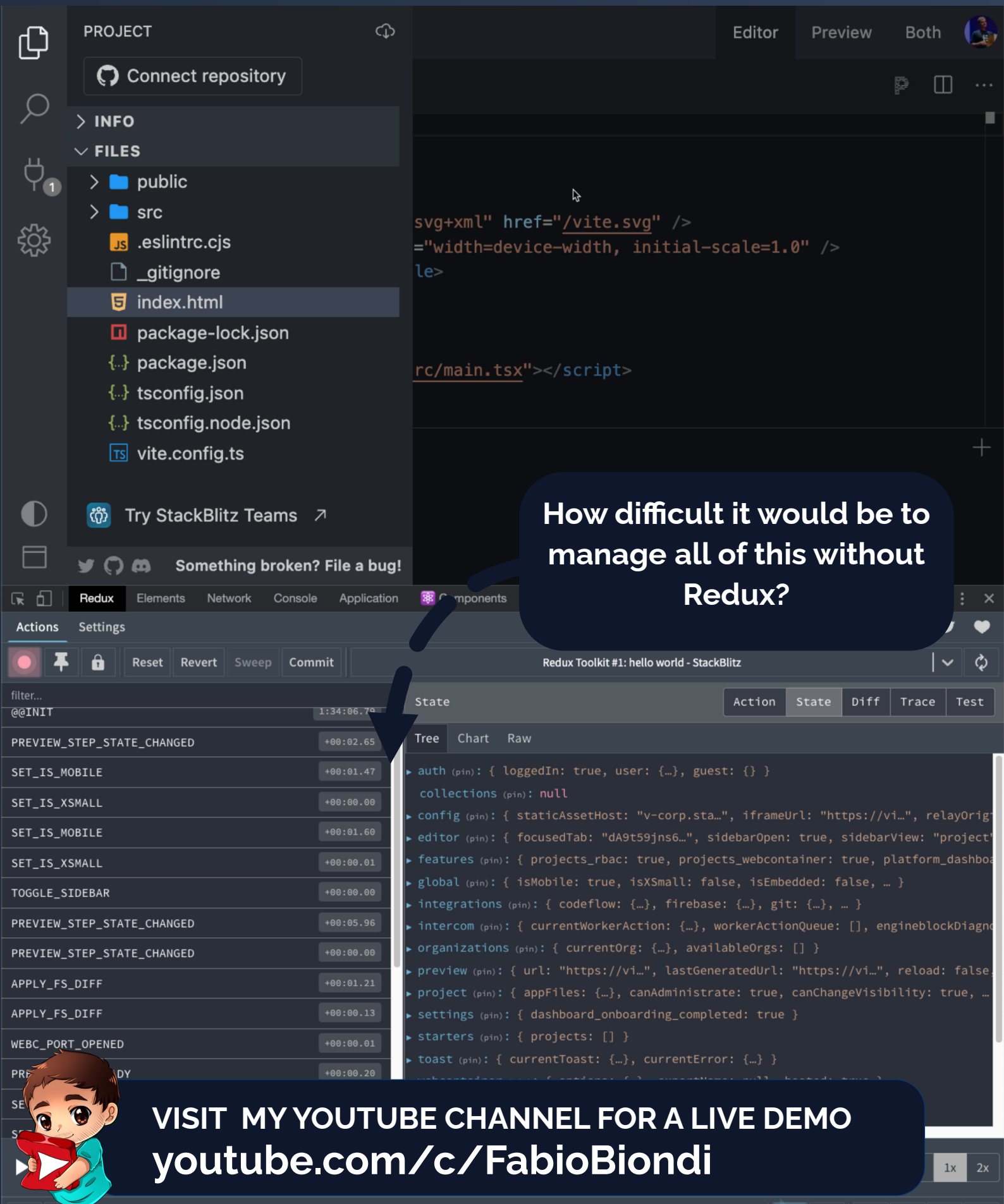


And how the state is changed
after an action has been dispatched



A real world example using Redux

StackBlitz



VISIT MY YOUTUBE CHANNEL FOR A LIVE DEMO

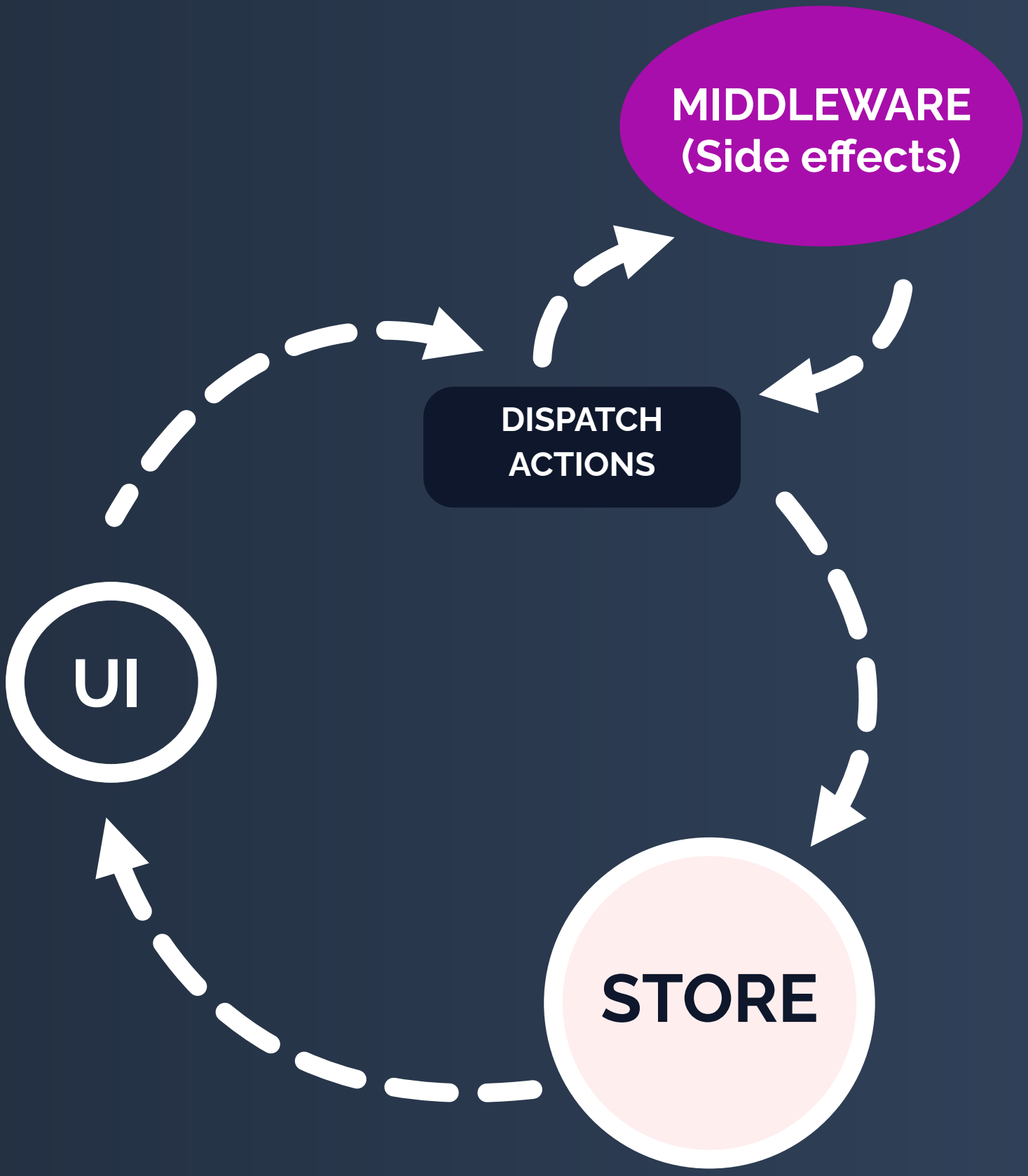
youtube.com/c/FabioBiondi

REDUX STORE

WHAT IS THE STORE?
The **store** is a global application state and it's composed by multiple reducers

WHAT IS A REDUCER?
It's a function that **initialize** and **mutate** a part of the store

REDUX FLOW



INSTALL DEPENDENCIES

Install Redux for React

```
npm i react-redux
```

it provides the **useSelector** and **useDispatch** hooks

Install Redux ToolKit (RTK)

```
npm i @reduxjs/toolkit
```

Include several utilities to simplify and enhance Redux

Configure the Store

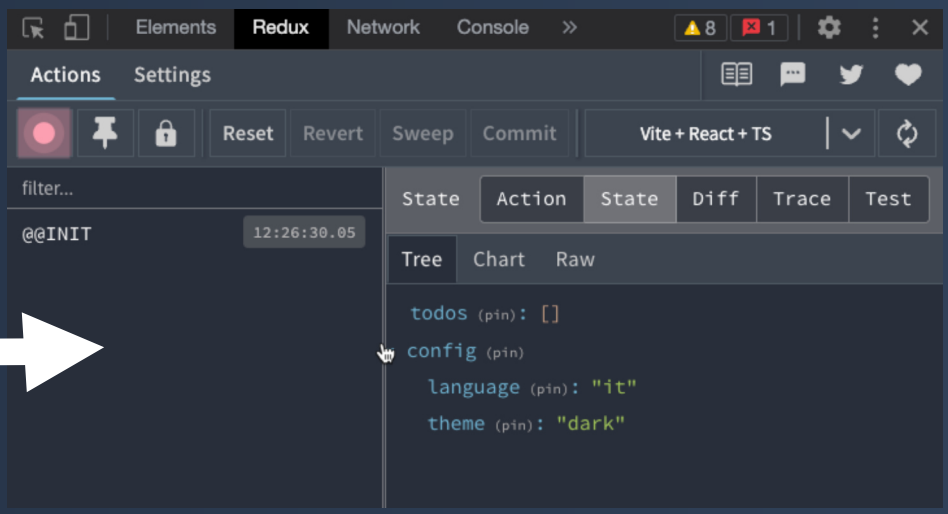
Store Creation

```ts
const rootReducer = combineReducers({
  todos: () => [],
  config: () => ({ theme: 'dark', lang: 'it'})
});

export const store = configureStore({
  reducer: rootReducer,
});

export type RootState = ReturnType<typeof rootReducer>;
```

Store Type



Wrap your app with Provider

```tsx
<Provider store={store}>
    <App />
</Provider>
```

ACTIONS

Payload Type

Action Type

counter.actions.ts

```ts
export const increment = createAction<number>('increment')
export const decrement = createAction<number>('decrement')
```

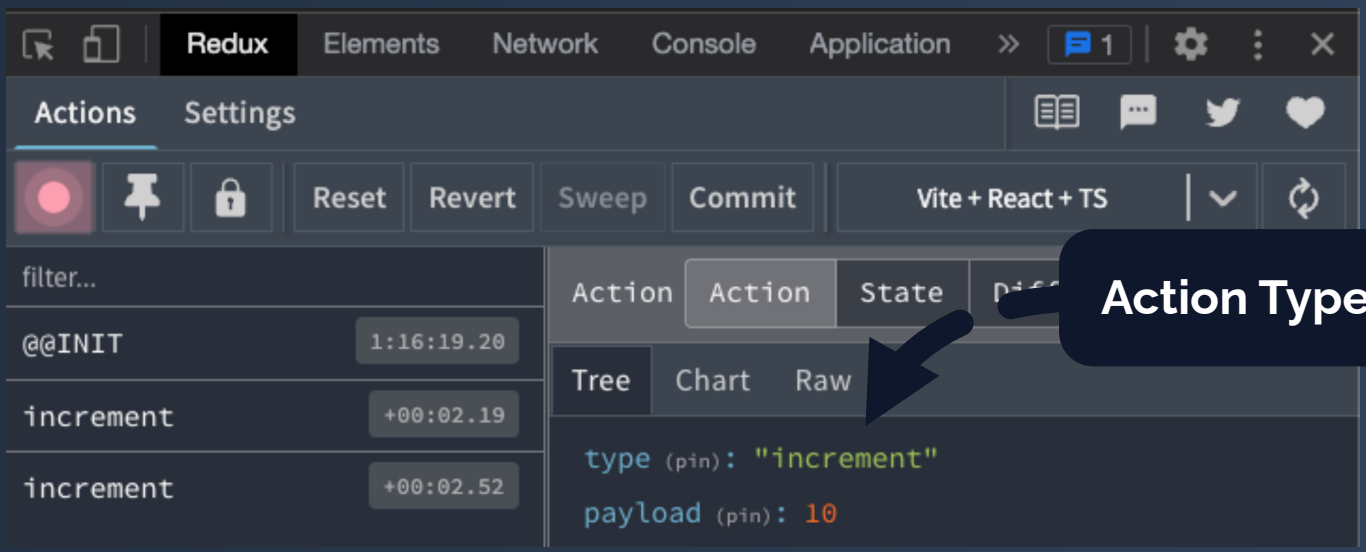
MyComponent.tsx

Dispatch Action

```tsx
export const Component = () => {
  const dispatch = useDispatch();

  return <div>
    <button onClick={() => dispatch(increment(10))}>+</button>
    <button onClick={() => dispatch(decrement(5))}>-</button>
  </div>
};
```

REDUX DEV TOOLS



Action Type

Payload

fabiobiondi.dev

REDUCER

WHAT IS A REDUCER?

It's a function that **initialize** and **mutate** a part of the store in according to the dispatched actions

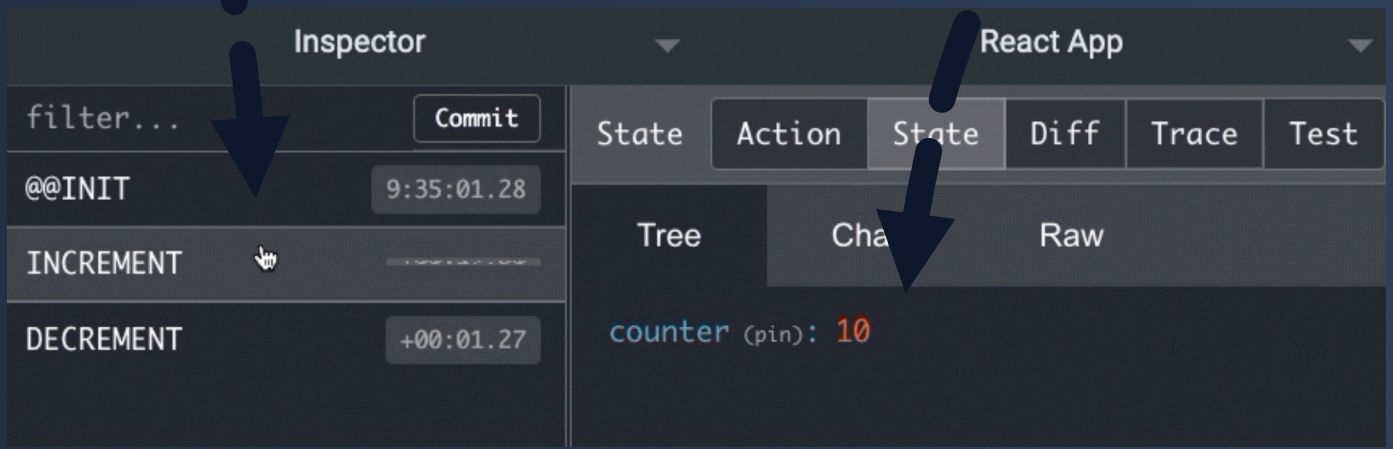
counter.reducer.ts

Initial State

```ts
export const counterReducer = createReducer(0, builder =>
  builder
    .addCase(increment, (state, action) => state + action.payload)
    .addCase(decrement, (state, action) => state - action.payload)
)
```

Actions

Update State



combineReducers

combines multiple reducers in one store

ADD REDUCERS to store

```
const rootReducer = combineReducers({
  counter: counterReducer,
  config: configReducer,
  todos: todoReducer,
  auth: authReducer,
  // ...
});
```

WHAT IS THE STORE?

The **store** is a global application state and it's composed by multiple reducers

createSlice

An utility to simplify the creation of reducers and actions in one place

(So it's an alternative to the previous approach)

```ts
const initialState: ConfigState = { language: 'it', theme: 'dark' };

export const configStore = createSlice({
  name: 'config',
  initialState,
  reducers: {
    changeTheme(state, action: PayloadAction<Theme>) {
      state.theme = action.payload;
    },
    changeLanguage(state, action: PayloadAction<Language>) {
      state.language = action.payload;
    },
  },
});

export const { changeLanguage, changeTheme } = configStore.actions;
```

ACTIONS

STATE UPDATE

EXPORT ACTIONS

HINT: IMMER JS

Redux does not allow to directly mutate the state but it's allowed in Redux Toolkit since it includes immerJS


```ts
const rootReducer = combineReducers({
  config: configStore.reducer
});
```

Add slice to Store

USAGE

Dispatch

```tsx
export const Component1 = () => {
  const dispatch = useDispatch();

  return (
    <div>
      <button onClick={() => dispatch(changeTheme('dark'))}>Dark</button>
      <button onClick={() => dispatch(changeLanguage('en'))}>English</button>
    </div>
  );
};
```

Selector

```tsx
export function Component2() {
  const theme = useSelector((state: RootState) => state.config.theme);
  const lang = useSelector((state: RootState) => state.config.language);

  return (
    <div>
      <div>Current THeme: {theme}</div>
      <div>Current Lang: {lang}</div>
    </div>
  );
}
```

But there is much more...

FOLLOW ME & LIKE
to get more info about
Front-End Development

VIDEO CORSO

REACT PRO

Creare Real World Applications

React 18, TypeScript, Zustand, Tailwind, Vite

160+ video

8+ ore di lezione

12 capitoli

italiano

ESTRATTO ARGOMENTI

- Creare shop con area riservata
- React & **TypeScript**
- Utilizzare **pochissime dipendenze**
- Creazione layout con **Tailwind**
- Custom Components e Custom hooks
- Local State con **useReducer**
- State Management con **Zustand**
- **Autenticazione** con JWT
- Good Practices, Clean Coding & Tips

v. 18.2

v. 4.2

TS v. 4.9

v. 4.3

DISPONIBILE SU

fabiobiondi.dev

fabio**biondi**.dev

youtube.com/c/FabioBiondi