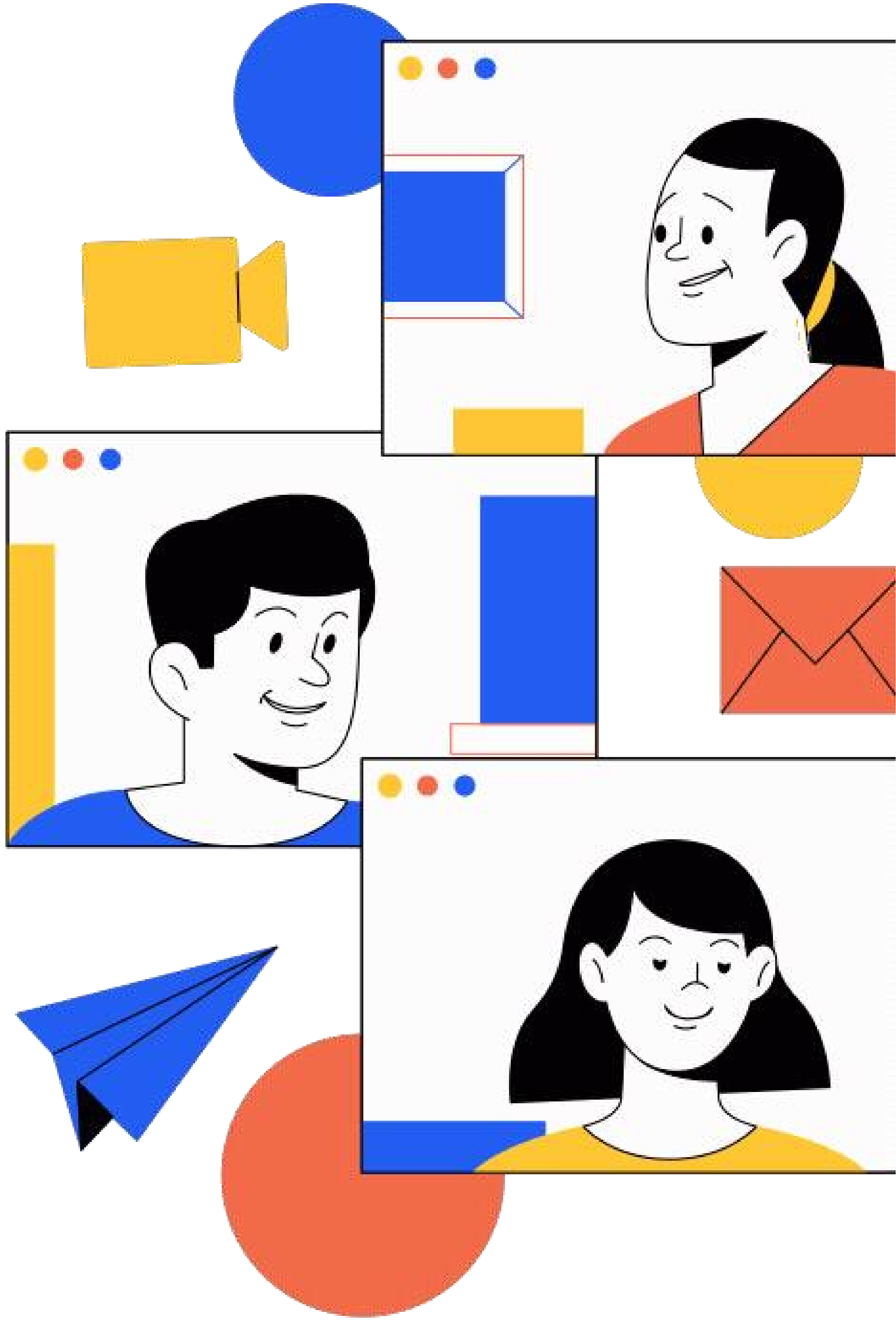
UDACITY

Week 4 - Redux

Today we are using different colors!



Ahmed Fouad Lotfy

React Session Lead

Agenda

What we'll cover in this session

- What is Redux?
- Why Redux?
- How to use Redux?
- Store, Actions, Reducers and Dispatch
- Live Demo

What is Redux?

- Redux is a JS library for predictable and maintainable global state management.
- Redux Toolkit wraps around the Redux core, and contains packages and functions for building a Redux app.
- It was created by Dan Abramov and Andrew Clark.
- Redux is an open source library.

Why choosing Redux?

- Single Source of Truth
- Predictable State Management
- Easy Debugging with DevTools
- Middleware for Extensibility
- Works Great with React

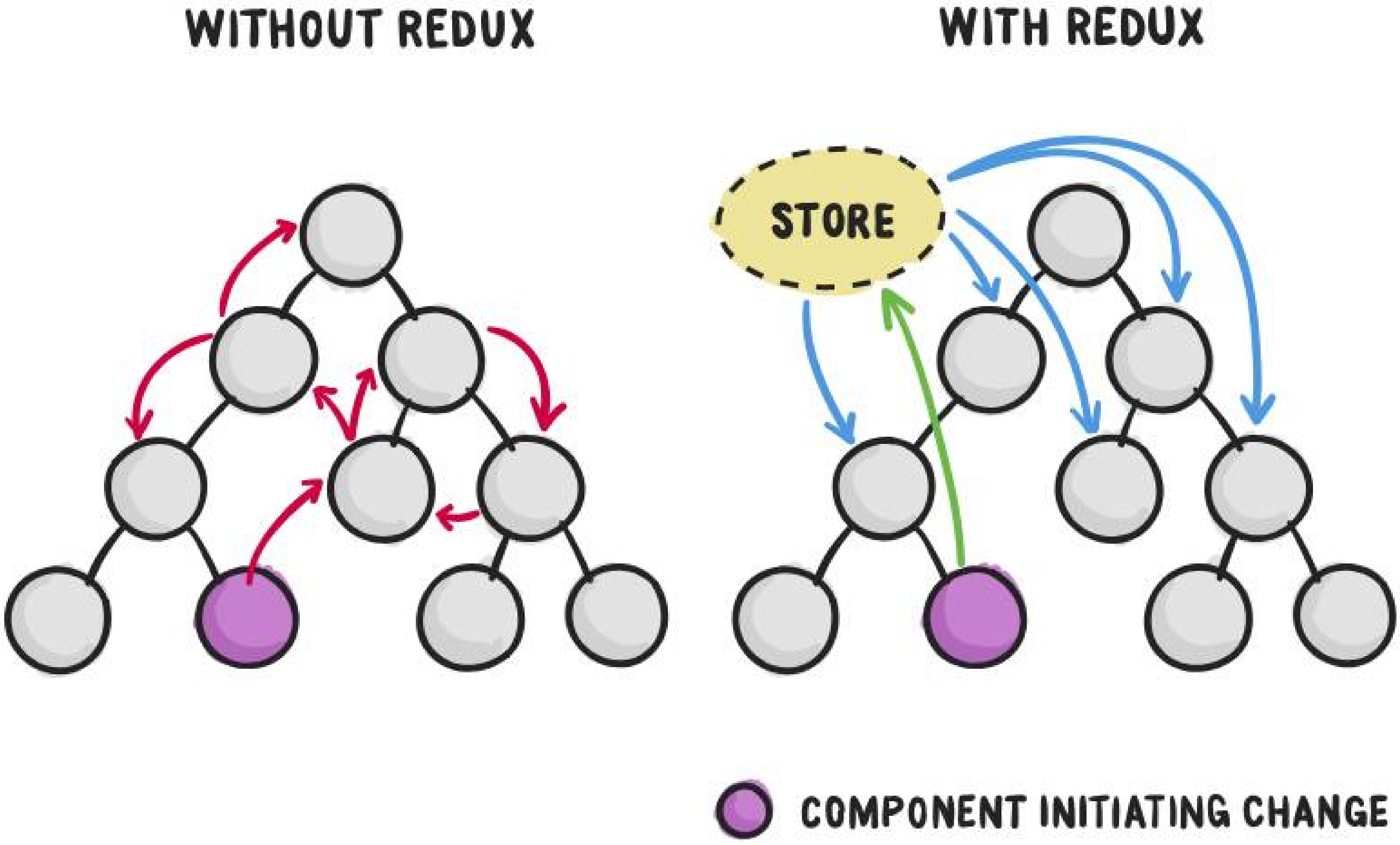
State Management

Distributed

- **State Location:** State is spread across multiple components, with each managing its own local state.
- **Data flow** can be bidirectional and fragmented, as each component updates its own state.
- **Debugging** requires tracking state across multiple components, which can be scattered and hard to trace.
- **Lower initial complexity** with simpler components, but becomes **harder** to manage as the application grows.

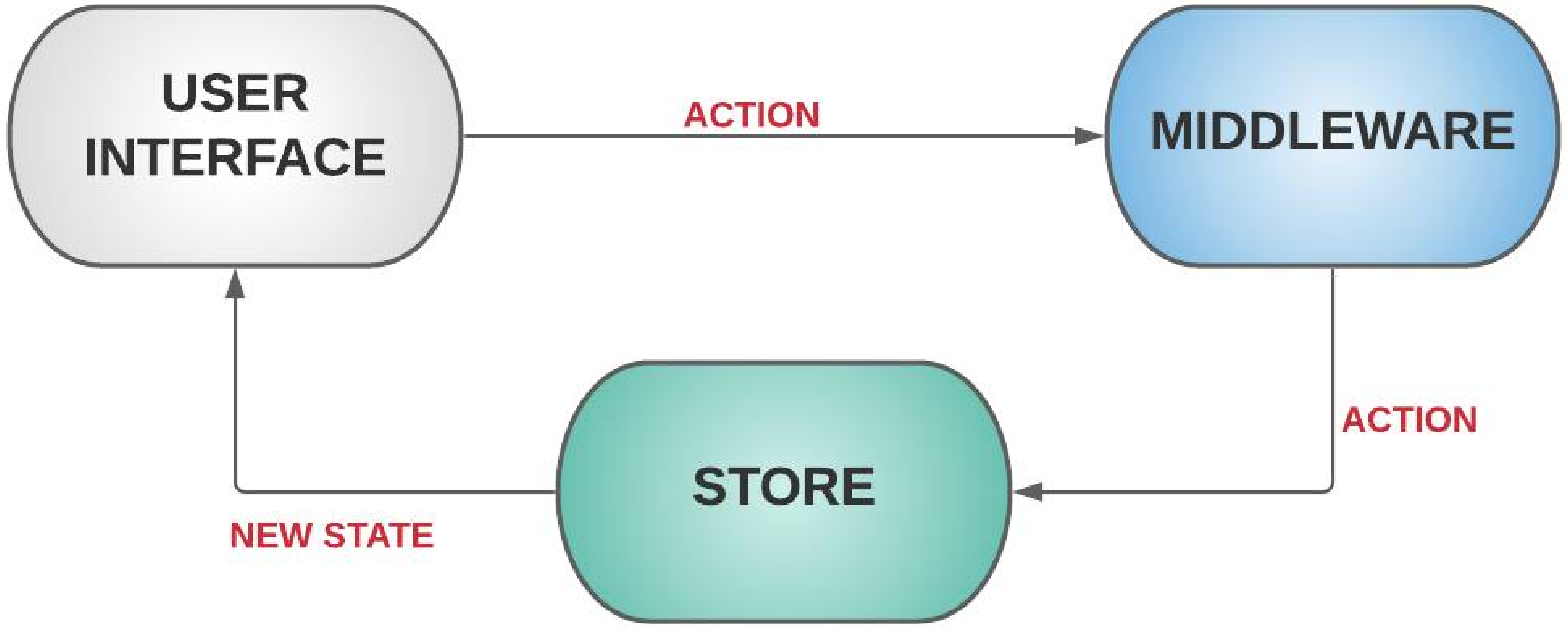
Centralized

- **State Location:** All state is stored in a single, central location (e.g., Redux store).
- **Data flows** in a unidirectional way: actions are dispatched, reducers update the state, and the updated state is reflected across the app.
- All state changes can be tracked and debugged in one place using tools like **Redux DevTools**.
- **Higher initial complexity** due to the setup of Redux or similar tools, but **scales well** as the app grows.


WITHOUT REDUX
WITH REDUX
STORE
● COMPONENT INITIATING CHANGE

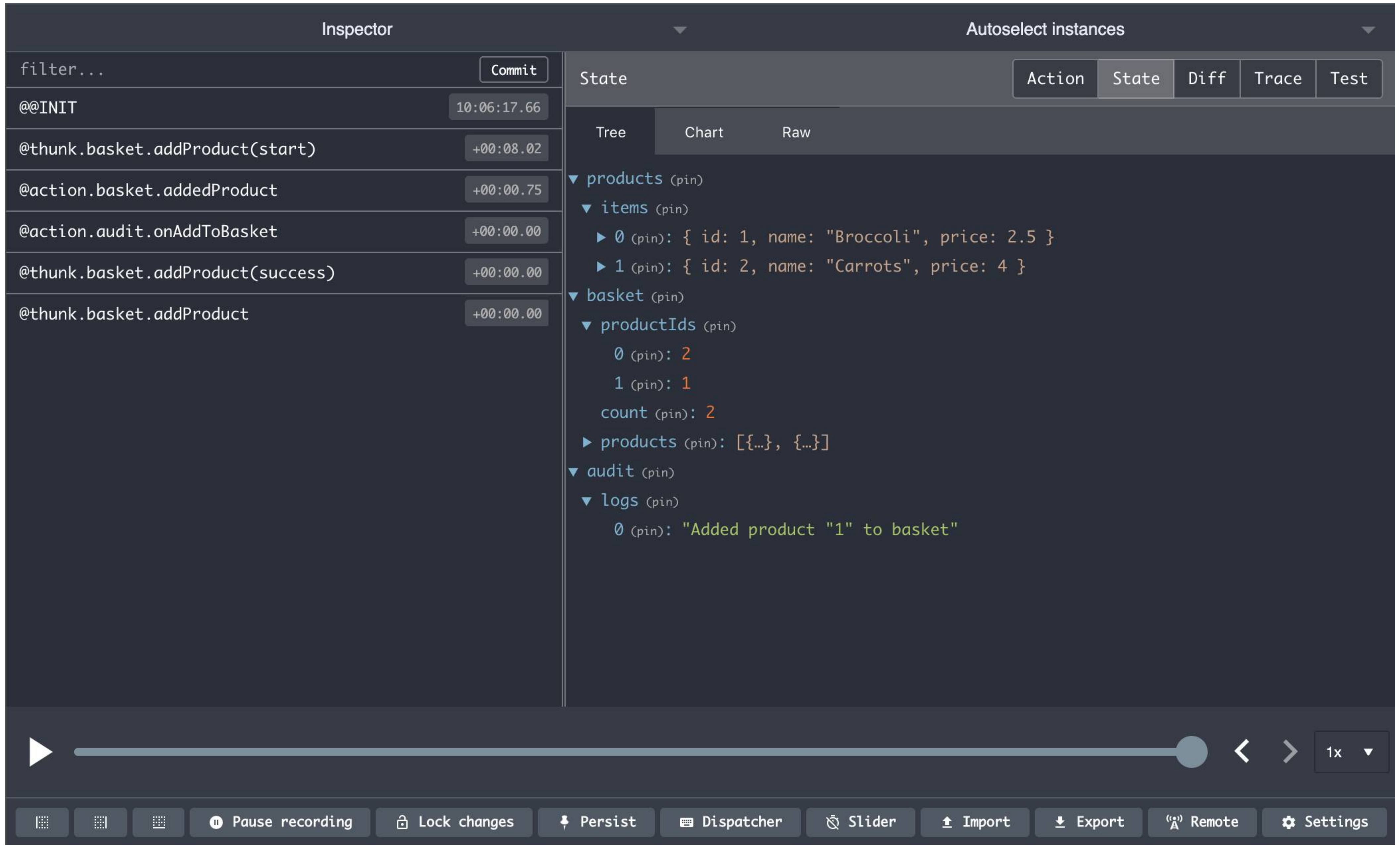
Predictable State Management

- State can only be changed by dispatching actions and processed by pure reducers.
- This predictability ensures that your application behaves consistently, making it easier to understand and maintain.



Easy Debugging with DevTools

- **Redux DevTools** Extension allows you to inspect every action and state change, time-travel to previous states, and even export/import state.
- These powerful debugging capabilities help you quickly identify and fix issues in your application.

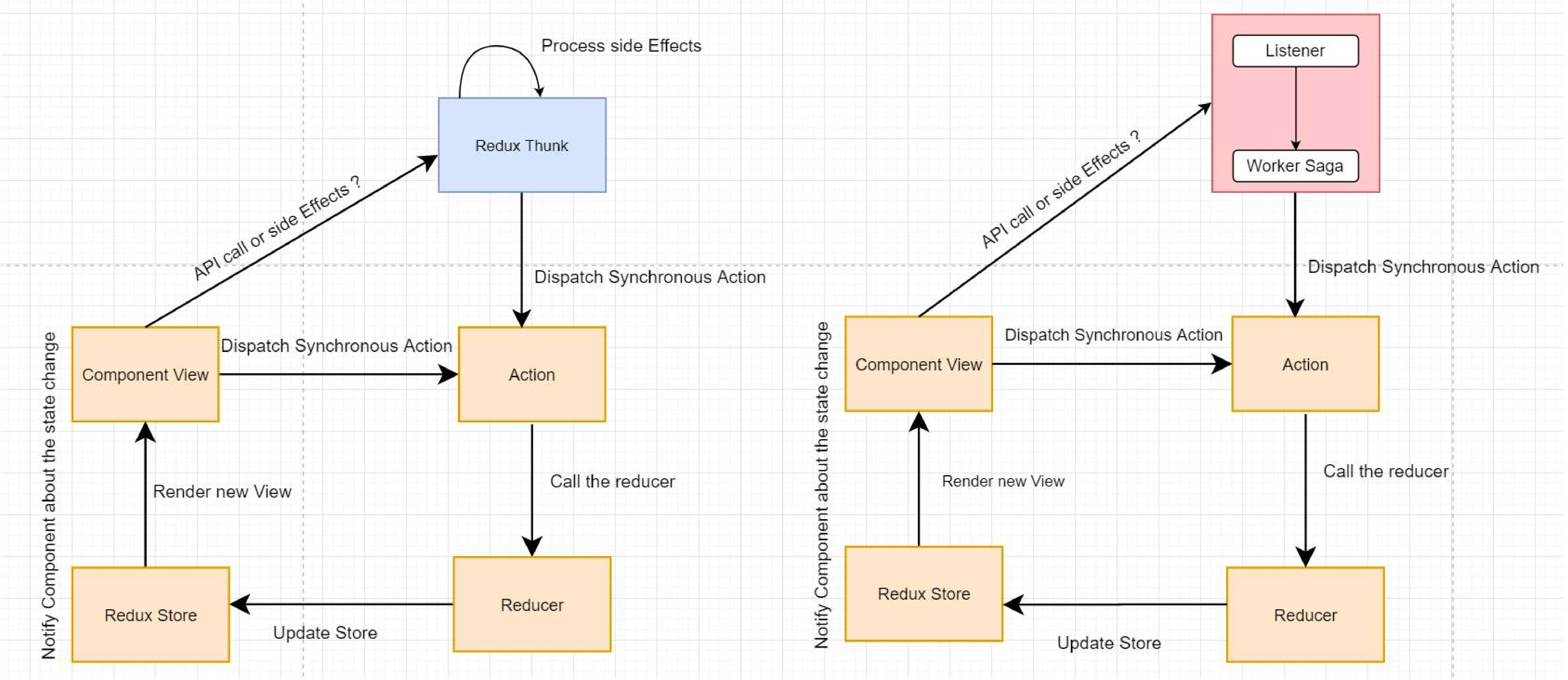


```
@@INIT                                  10:06:17.66
@thunk.basket.addProduct(start)         +00:08.02
@action.basket.addedProduct             +00:00.75
@action.audit.onAddToBasket             +00:00.00
@thunk.basket.addProduct(success)       +00:00.00
@thunk.basket.addProduct                +00:00.00
```

```
▼ products (pin)
 ▼ items (pin)
   ▶ 0 (pin): { id: 1, name: "Broccoli", price: 2.5 }
   ▶ 1 (pin): { id: 2, name: "Carrots", price: 4 }
▼ basket (pin)
 ▼ productIds (pin)
    0 (pin): 2
    1 (pin): 1
   count (pin): 2
 ▶ products (pin): [{…}, {…}]
▼ audit (pin)
 ▼ logs (pin)
    0 (pin): "Added product "1" to basket"
```

Middleware for Extensibility

- **Middleware in Redux** allows you to intercept actions and perform additional tasks, such as handling asynchronous operations (e.g., Redux Thunk or Redux Saga)..
- This extensibility makes it easier to handle complex scenarios like API calls or logging.



Process side Effects
Redux Thunk
API call or side Effects ?
Dispatch Synchronous Action
Component View
Dispatch Synchronous Action
Action
Notify Component about the state change
Render new View
Call the reducer
Redux Store
Update Store
Reducer
Listener
Worker Saga
API call or side Effects ?
Dispatch Synchronous Action
Component View
Dispatch Synchronous Action
Action
Notify Component about the state change
Render new View
Call the reducer
Redux Store
Update Store
Reducer

How to use Redux?

1) Install Redux Toolkit

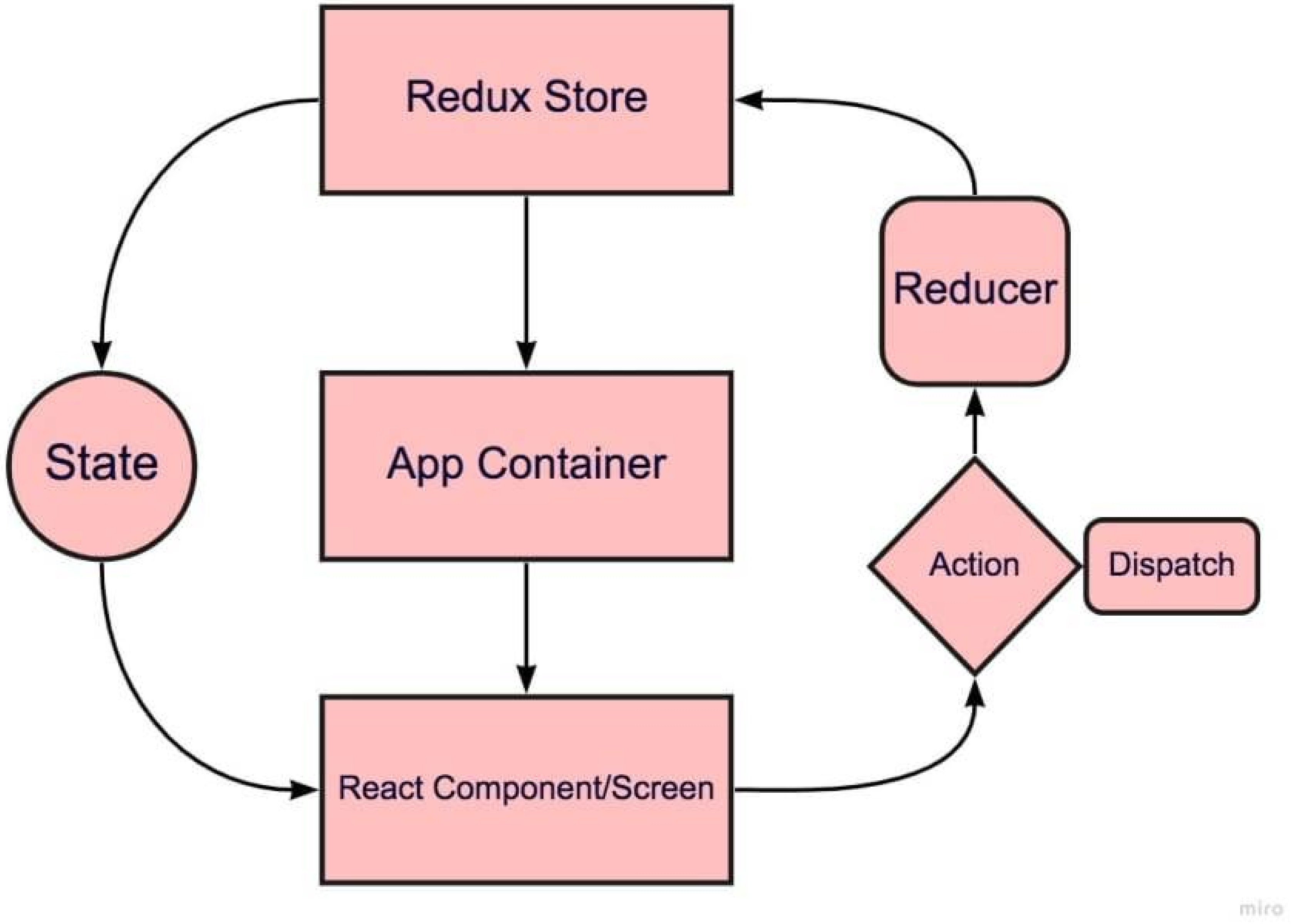
2) Define Actions types

3) Create Action Creators

4) Define the Initial State

5) Create Reducers

6) Create the Redux Store



Redux
Main Components

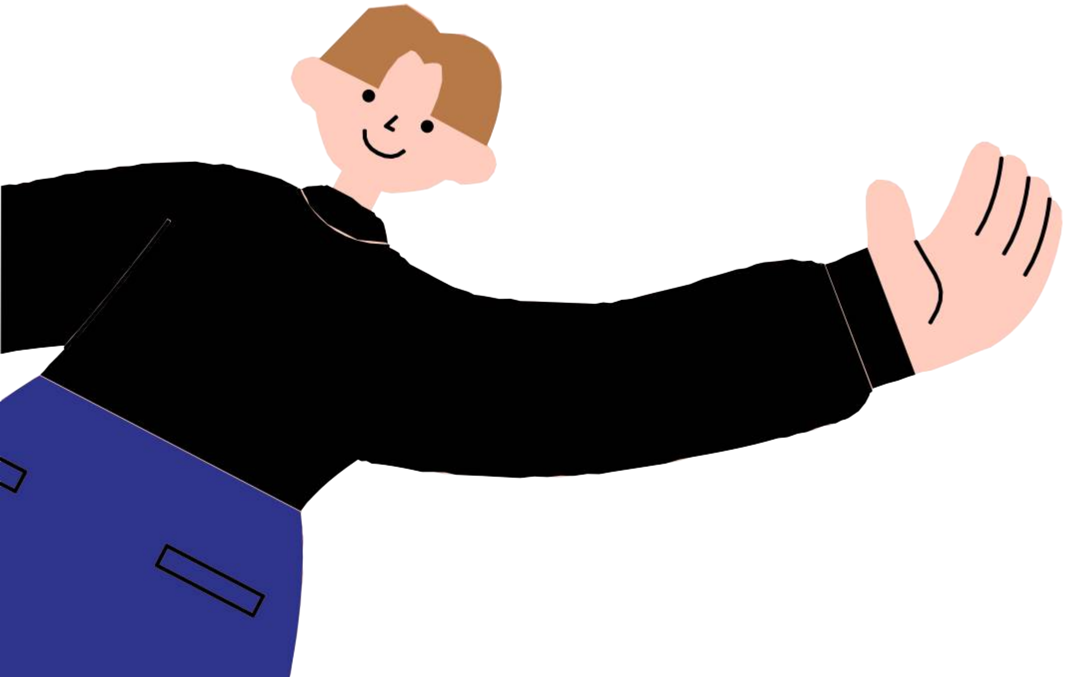
Redux has 4 different main components: **Store**, **Actions**, **Reducers**, and **Dispatch**

- **Store** is the central database that holds the entire state of your application. It is an object that contains the state and provides methods to access and manipulate that state.
- **Actions** are like instructions or commands that you send to the store. They tell the store what kind of change you want to make to the state.
- **Reducers** are pure functions that take the current state and an action as arguments and return a new state. They specify how the state changes in response to an action.
- **Dispatch** When you want to change the state, you use dispatch to send an action. The store then forwards this action to the reducer, which processes it and returns a new state.

Its Demo Time

Grab your coffee and let's get started. ☕

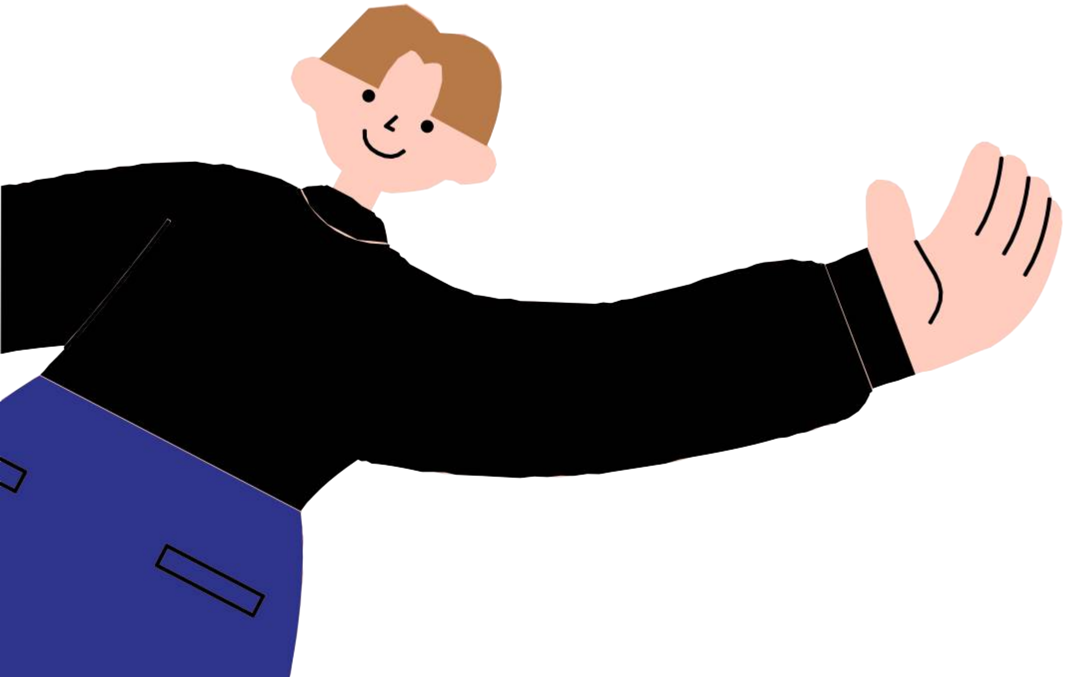
Project Checkpoint

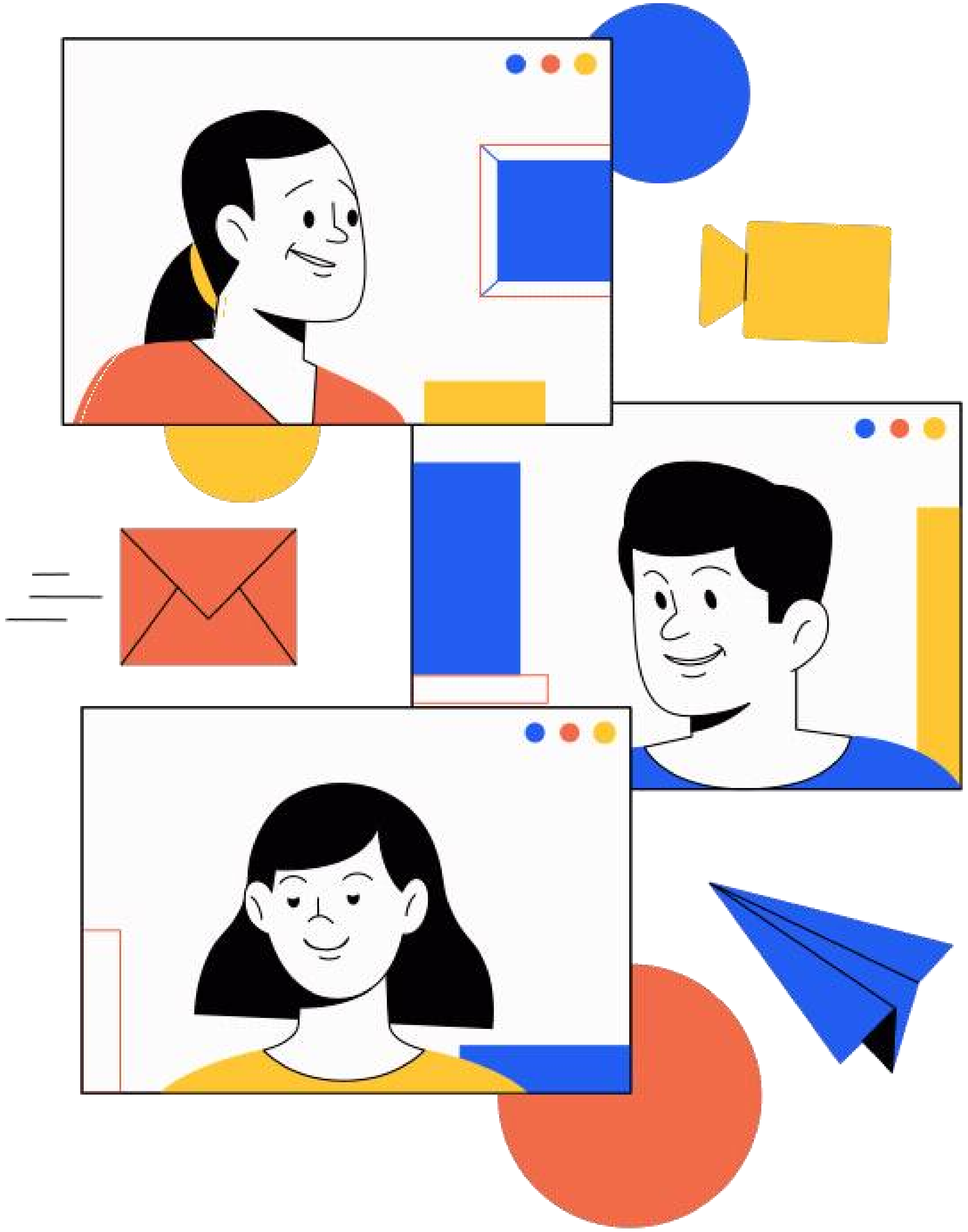


Next Session:

More about Redux

Any Questions?





Thank you for attending!

Feel free to email at **a.lotfy@fci-cu.edu.eg** or reach me at circle anytime for any questions or clarifications!

Follow me on Github **@ahmeddxfouad**

code and slides are found at this **github_repo**