Node FSD Training

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com

Week-6: Schedule for NodeJS FSD Training

Day#	Date	Topic
Day-26	15-May-2024	React JS : Forms, Class Components, Hooks
Day-27	**16-May-2024**	**Redux, DevOps CI/D & Docker**
Day-28	20-May-2024	Azure
Day-29	21-May-2024	Case Study Frontend + Backend Development + Integration + Unit Testing - 4+Technical discussion
Day-30	22-May-2024	Case Study Frontend + Backend Development + Integration + Unit Testing - 4+Technical discussion

State Management using Redux

React

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com

What is State management?

Narasimha

Sr. IT Trainer/Consultant

Statement Management

- React applications are built using components and they manage their state internally and it works well for applications with few components, but when the application grows bigger, the complexity of managing states shared across components becomes difficult.
- For example consider an e-commerce application, in which the status of multiple components will change when purchasing a product.

Statement Management -- Case Study

1. Add that product to the shopping list
2. Add product to customer history
3. Trigger count of purchased products
4. If developers do not have scalability in mind then it is really hard to find out what is happening when something goes wrong.

This is why you need state management in your application.

State management Libraires

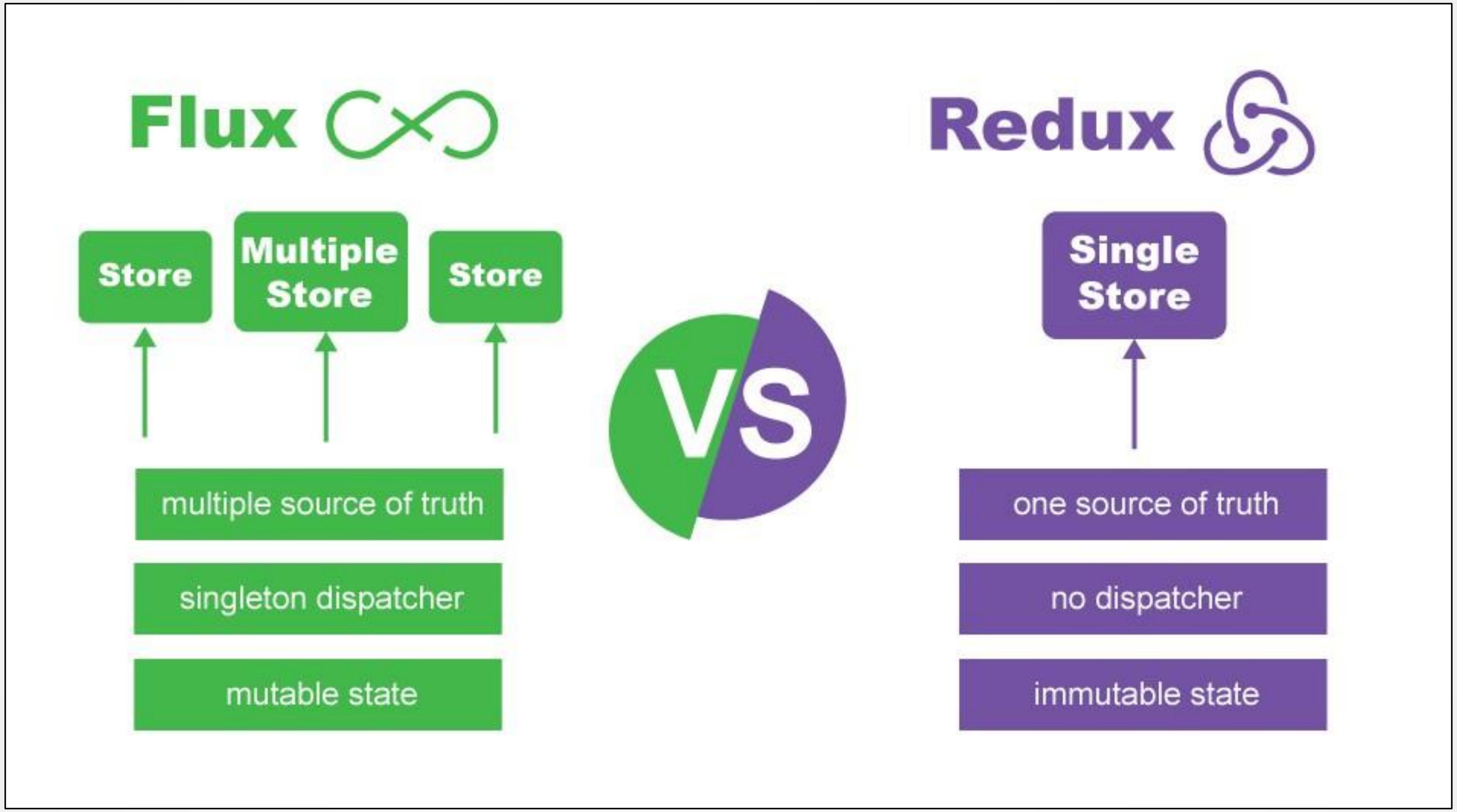
Narasimha

Sr. IT Trainer/Consultant

State management Libraires

1. Flux
2. Redux
3. MobX
4. Akita (from salesforce)

Flux vs Redux



Introduction to Redux

Narasimha

Sr. IT Trainer/Consultant

What is Redux?

- Redux is a pattern and library for managing and updating application state, using events called “actions”.
- It serves as a centralized store for state that needs to be used across your entire application, with rules ensuring that the state can only be updated in a predictable fashion.

Introduction to Redux

- Redux is an open-source JavaScript library
- It is used for managing application state.
- It is most commonly used with libraries such as React or Angular for building user interfaces.
- Similar to Facebook's Flux architecture
- It was created by Dan Abramov and Andrew Clark.

A few things to consider before using redux

- You have large amounts of application state that are needed in many places in the app
- The app state is updated frequently
- The logic to update that state may be complex
- The app has a medium or large-sized codebase, and might be worked on by many people
- You need to see how that state is being updated over time

Key Players in Redux (Store, Reducers, Actions)

React

Narasimha

Sr. IT Trainer/Consultant

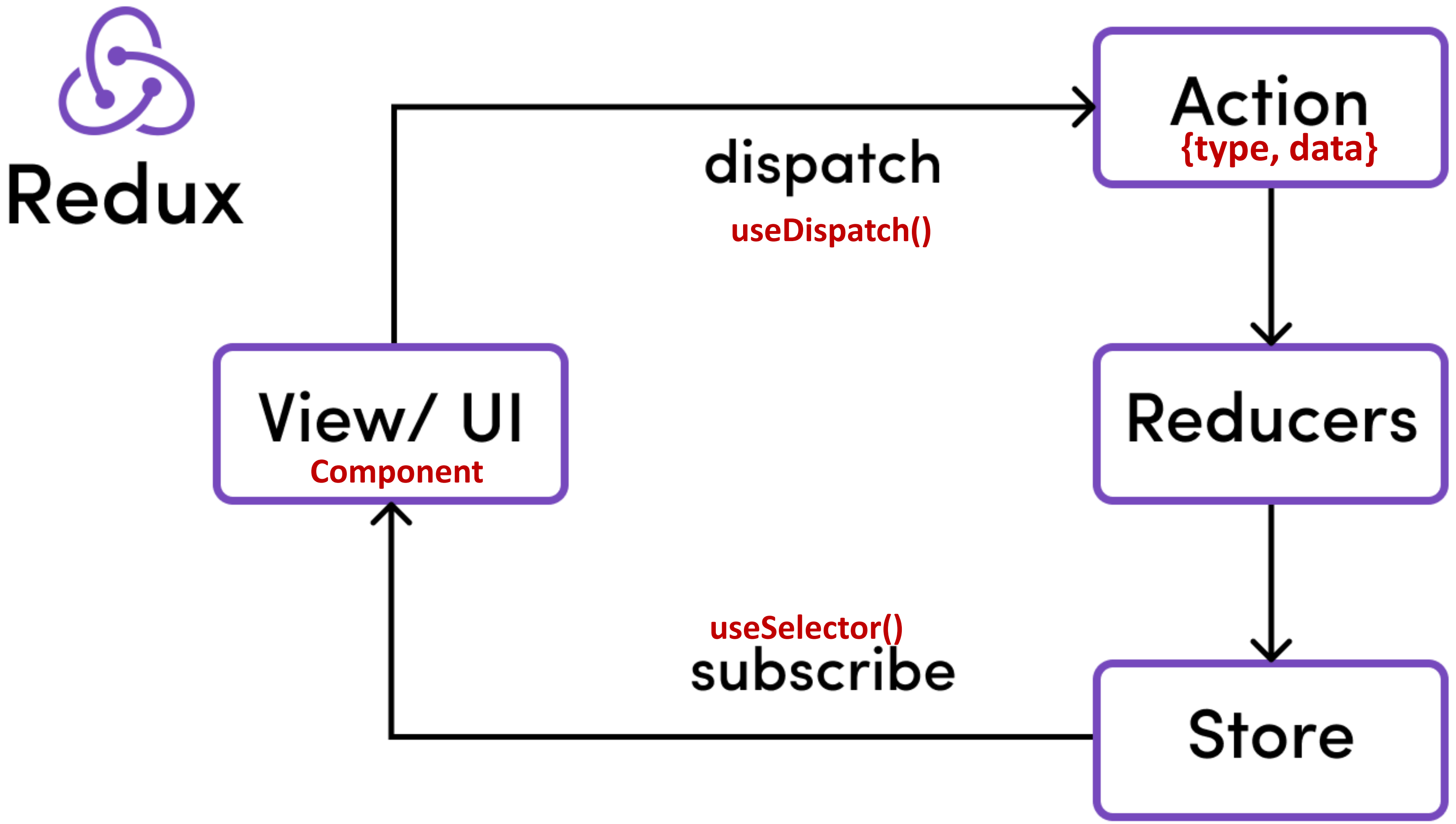
Exploring The Core Redux Concepts

1. Store
2. Actions
3. Reducers

1. **Store**: it brings the actions and reducers together, holding and changing the state for the whole app — there is only one store.

2. **Actions**: An object that have two properties, one describing the **type of action,** and one describing what should be changed in the app state.

3. **Reducers**: Functions that implement the behavior of the actions. They change the state of the app, based on the action.


Redux
dispatch
useDispatch()
Action
{type, data}
View/ UI
Component
Reducers
useSelector()
subscribe
Store

Environment Setup

1. npm i redux
2. npm i react-redux

Redux Implementation

Narasimha

Sr. IT Trainer/Consultant

Steps

1. Create reducer function [bankStore.js]
2. Create Store using redux library with reducer function [bankStore.js]
3. Use <Provider/> to share the store to components [index.js]
4. Use Redux hooks to communicate with store: **useSelector, useDispatch** [BankApp.js]
5. Perform the operations using dispatch [BankApp.js]

Steps1 & 2 : Create Reducer and Store

```
import { legacy_createStore as createStore } from 'redux';

// Reducer Function
const bankReducer = (state, action) =>
{


};

// Create Store
const bankStore = createStore(bankReducer);
export default bankStore;
```

Steps3: Share the store using Provider (index.js)

```
<Provider store={bankStore}>
            <BankApp  />
</Provider>
```

Steps4_5 : Communicate with store in Components

```
import { useSelector, useDispatch } from "react-redux";

function BankApp() {
    const [amountValue, setAmount] = useState(0);
    let currentBalance = useSelector((state) => state.balance);
    const dispatch = useDispatch();

    function deposit_click() {
        dispatch({type:"DEPOSIT", amount:amountValue} );
        setAmount(0);   // clear textbox
    }
}
```

Practice Hands-Ons

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com