React JS Training

(Intermediate to Advanced)

React

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com

Schedule for React JS Training

Day#	Date	Topic
Day-1	16-Apr-2024	State Management : Class Components
Day-2	17-Apr-2024	State Management : Functional Components, Hooks
Day-3	18-Apr-2024	Http Client Programming – AJAX Calls to APIs (Node JS)
Day-4	19-Apr-2024	Working with Forms, Validations, Services
Day-5	22-Apr-2024	Redux – State Management Library
Day-6	23-Apr-2024	Unit Testing React Application
Day-7	24-Apr-2024	Routing – SPA in React JS

Day5
React-Redux

React

Narasimha
Sr. Corporate Trainer, Mentor
tnrao.trainer@gmail.com

Index – Day5

1. State management in React Appliations
2. Introduction to Redux
3. Key Players in Redux : Store, Action, Reducers
4. ReactJS - Redux concepts with examples
5. Data Sharing using React Context

What is State management?

Narasimha

Sr. IT Trainer/Consultant

Statement Management

1. 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.
2. 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.

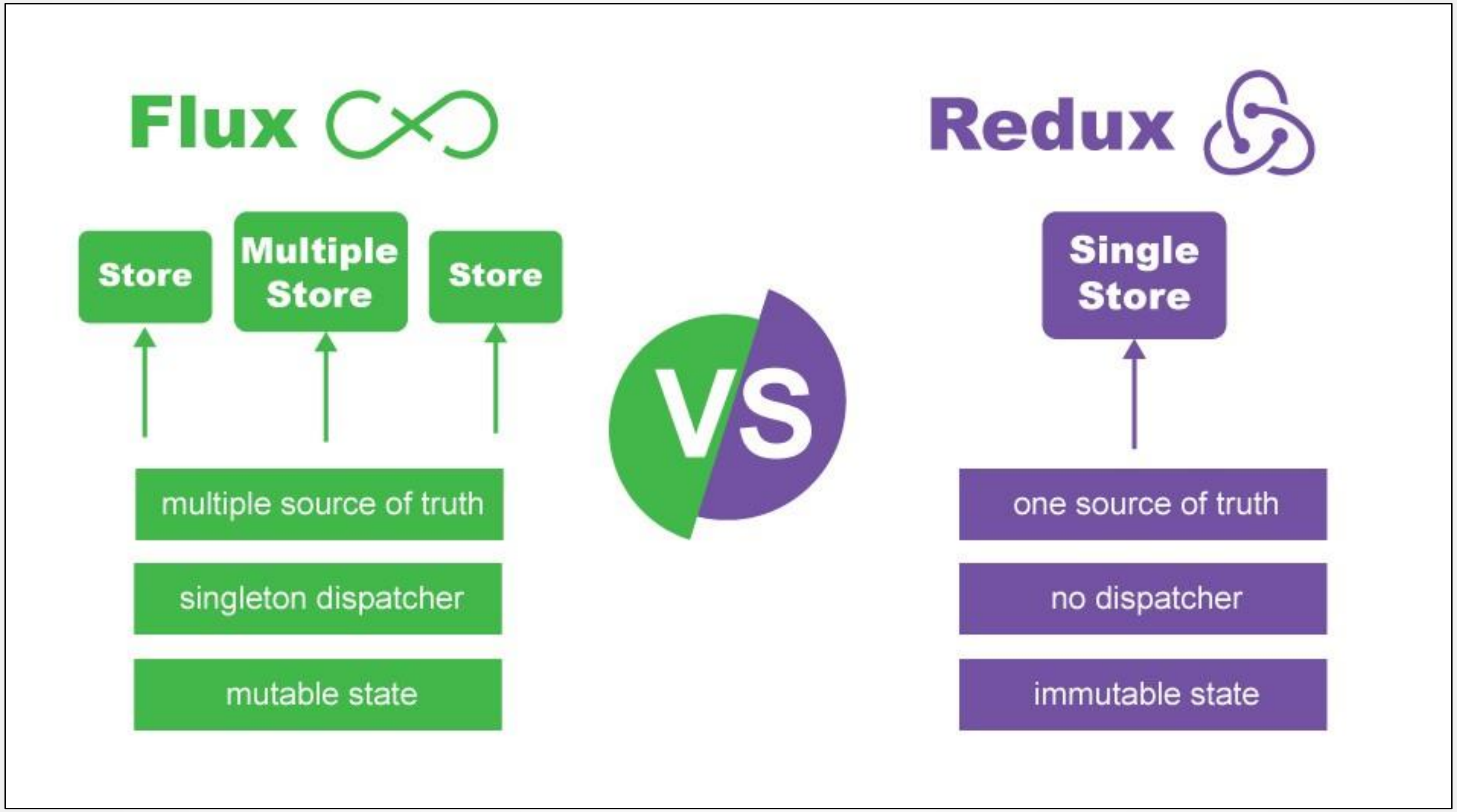
This is why you need state management in your application.

State management Libraires

Narasimha

Sr. IT Trainer/Consultant

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

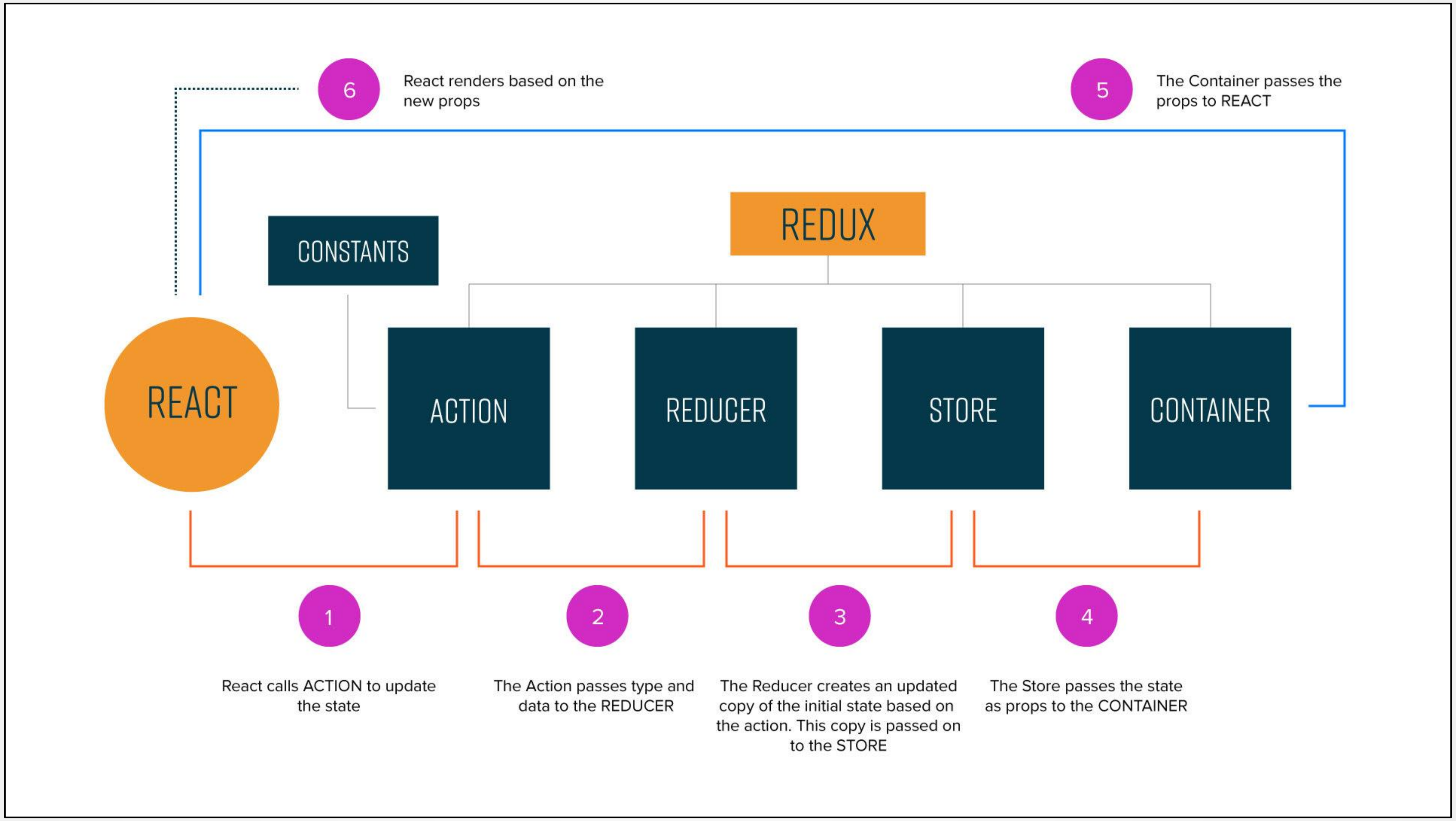
1. Store
2. Actions
3. Reducers

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

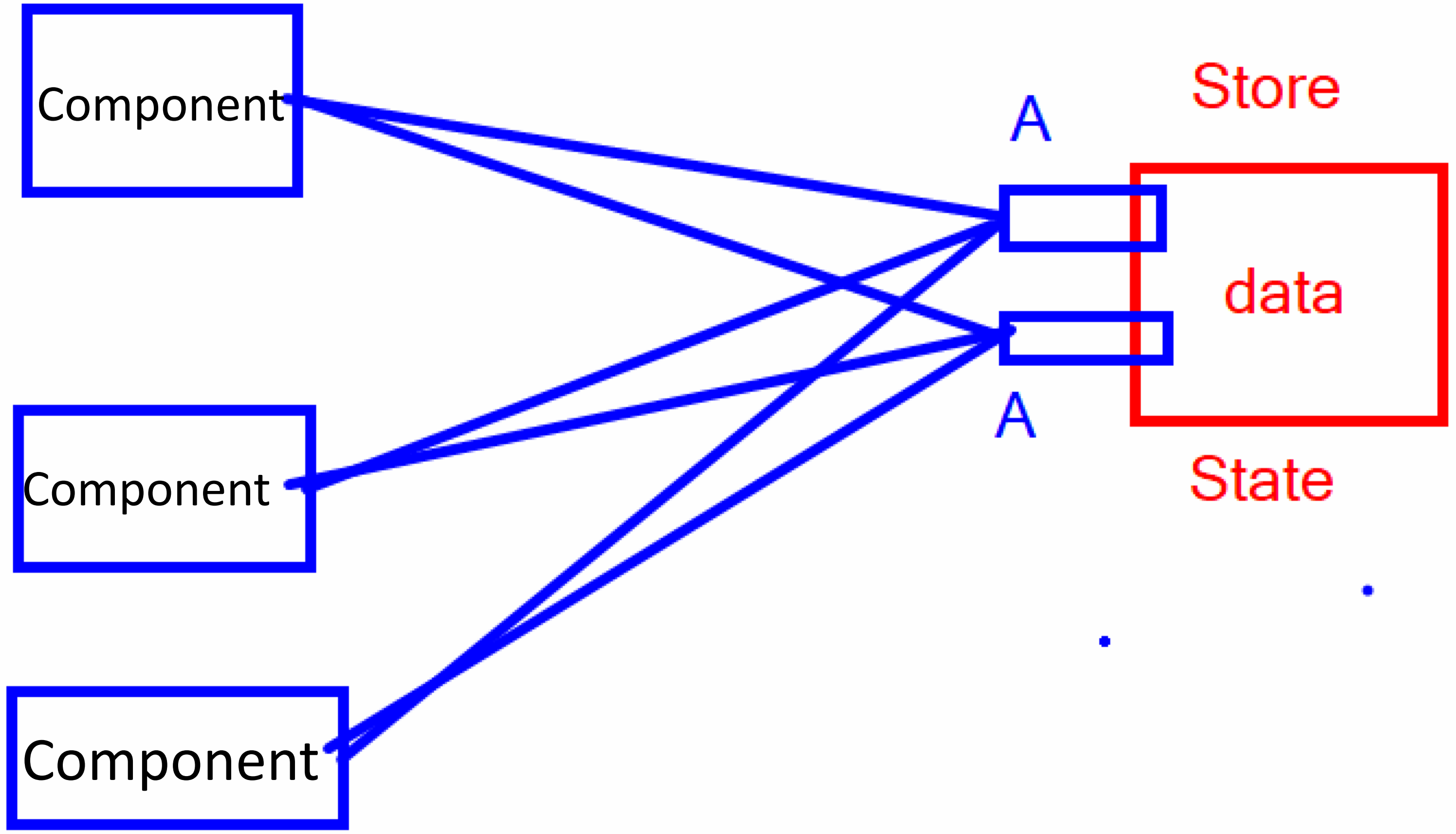
2. **Actions**: An object that have two properties, one describing the **type of action,** and one describing what should be changed in the app state.

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

How Redux Works?



6
React renders based on the new props
5
The Container passes the props to REACT
REDUX
CONSTANTS
REACT
ACTION
REDUCER
STORE
CONTAINER
1
2
3
4
React calls ACTION to update the state
The Action passes type and data to the REDUCER
The Reducer creates an updated copy of the initial state based on the action. This copy is passed on to the STORE
The Store passes the state as props to the CONTAINER


Component
Component
Component
A
A
Store
data
State

Environment Setup

1. npm i redux   --save
2. npm i react-redux   --save
3. npm i @reduxjs/toolkit

Redux Implementation

Narasimha

Sr. IT Trainer/Consultant

Steps

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

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)

```
import { Provider } from 'react-redux';
import bankStore from './Stores/BankStore’;

bankStore.dispatch({type:"CREATE"});

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

Steps4_5 : Communicate with store in Components

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

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

Steps4_5 : Communicate with store in Components

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

function BankApp()
{
 ………
 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

Sharing Global data using React Context

Narasimha

Sr. IT Trainer/Consultant

React Context

- React Context is a way to manage state globally.
- It can be used together with the useState Hook to share state between deeply nested components more easily than with useState alone.
- useContext is a React Hook that lets you read and subscribe to context from your component.

Steps to implement React Context

1. Create Context

   `const UserContext = createContext()`

2. Context Provider

   `<UserContext.Provider value={user}> </UserContext>`

3. Use the useContext Hook

   `const user = useContext(UserContext);`

Practice Hands-Ons

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com