Redux

State of the App

Roadmap

1. Problem
2. Solution
3. What is Redux?
4. Principles of Redux
5. Flow of a React-Redux application

The Problems

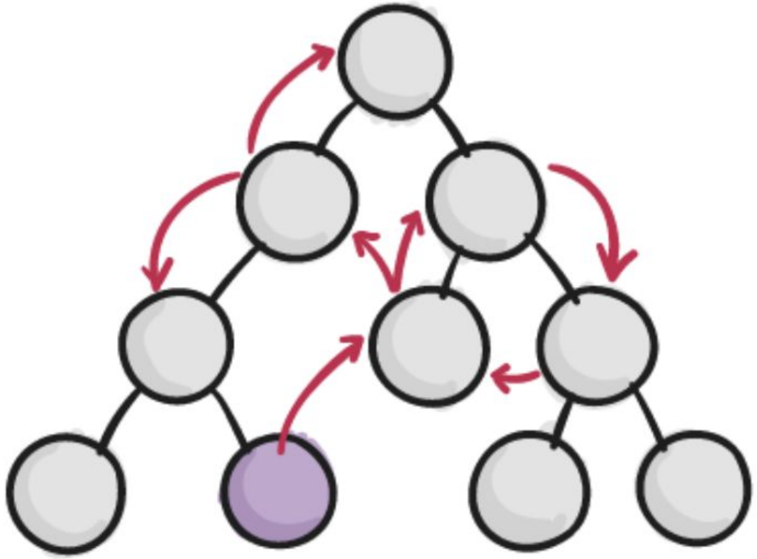
Ever-changing state

Everything is connected to EVERYTHING ELSE

If we make a change somewhere, something else will break

Passing tons of props, needless rerendering

WITHOUT REDUX



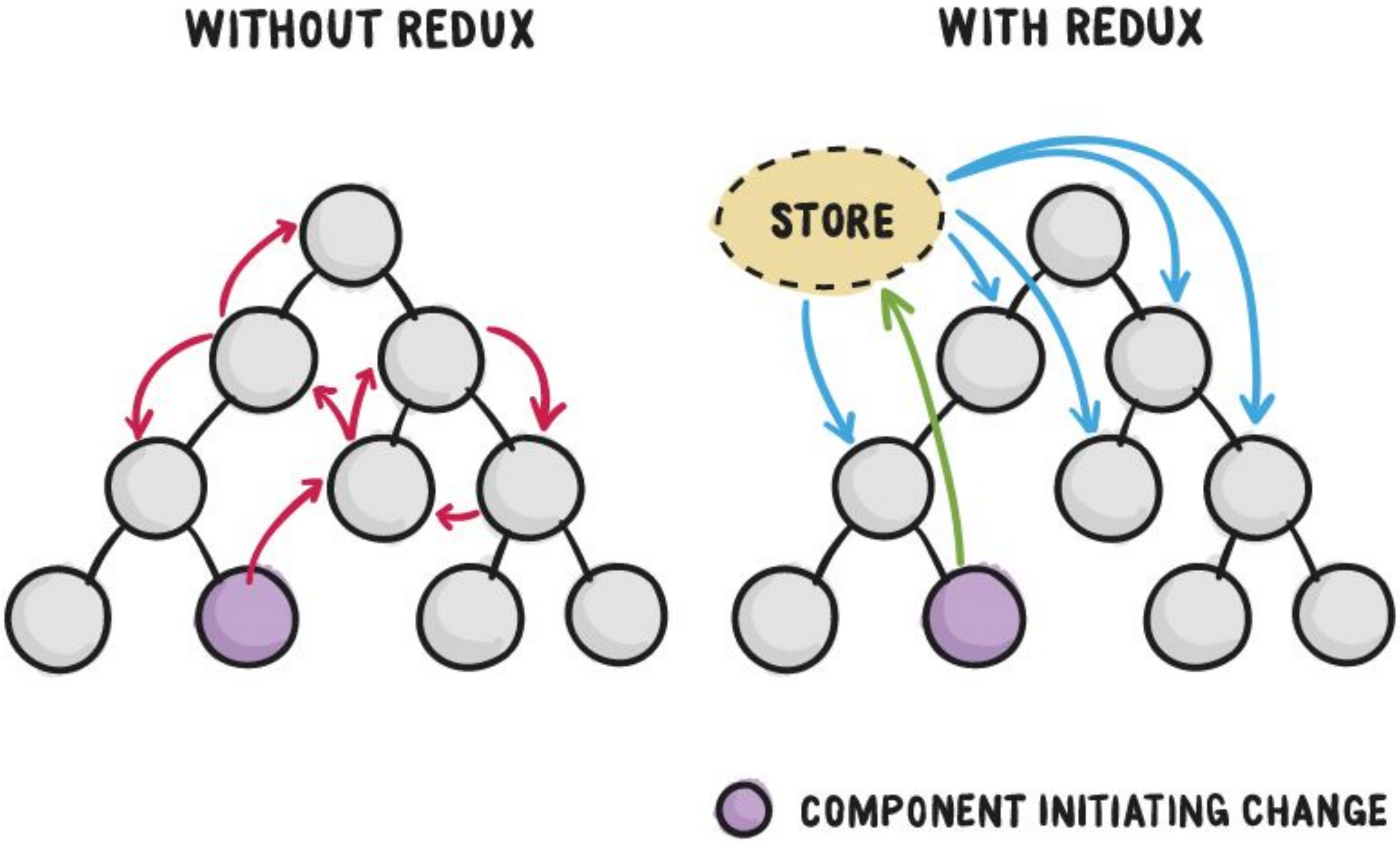
Solution: Redux

What is Redux?

- A tiny library
- A design pattern

What is the goal of Redux?

Make state changes predictable.


WITHOUT REDUX
WITH REDUX
STORE
COMPONENT INITIATING CHANGE

Principles of Redux

1. Single Source of Truth
2. Read-Only
3. Pure Functions
4. Unidirectional Flow

1. Single Source of Truth

The state of your whole application is stored in an object tree within a single store.

Store

An object that has methods such as getState() and dispatch(). It is the gatekeeper for access and alterations to state.

There is only one store for a redux app.

2. Read-Only

The state never changes.

The store is alerted of changes, and then based on that previous state, a new state is returned.

The only way to change state is to dispatch an action.

Actions

An action is a plain object containing the instructions and information that describes the state change we expect to see.

An action is an object with two keys:

1. Type: the command describing the state change
2. Payload: any data needed to complete the state change

~~function → state~~

---

component → action → reducer → store

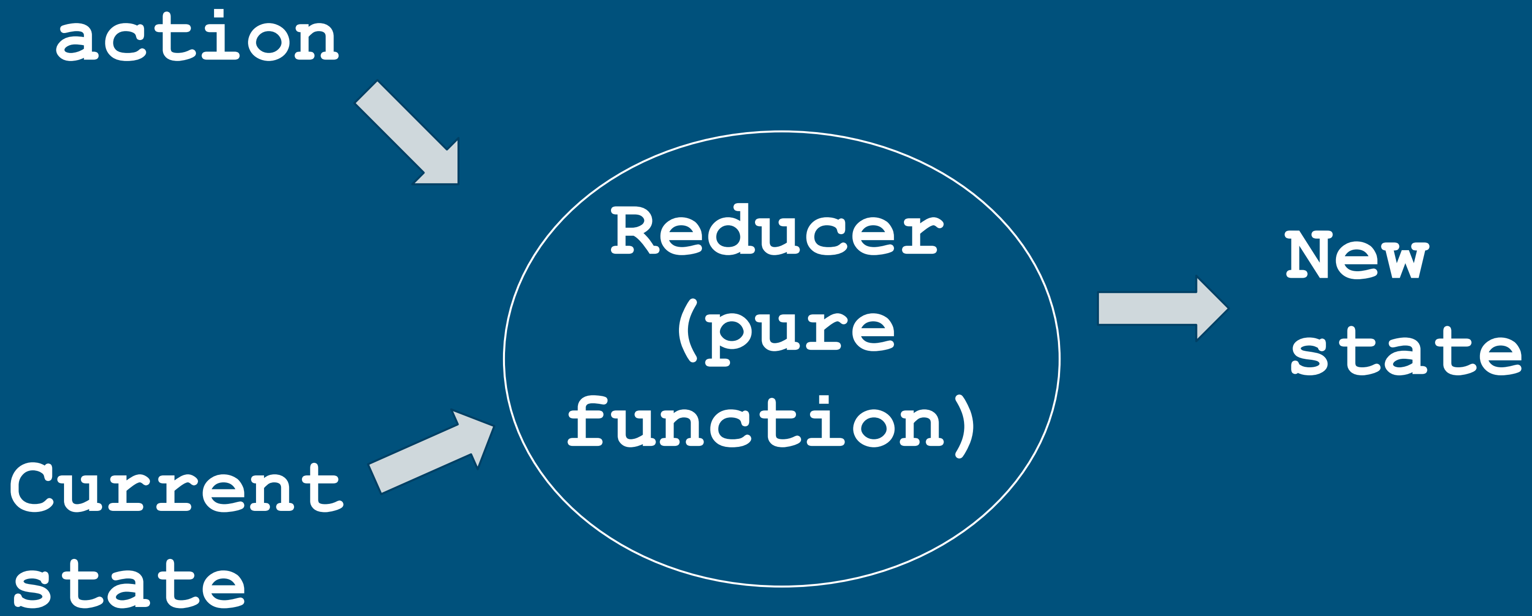
3. Pure Functions

When we get an action telling us how the state should change, we use pure functions that utilize pass by reference in our reducers to return a new state, not mutate the existing state.

Reducers

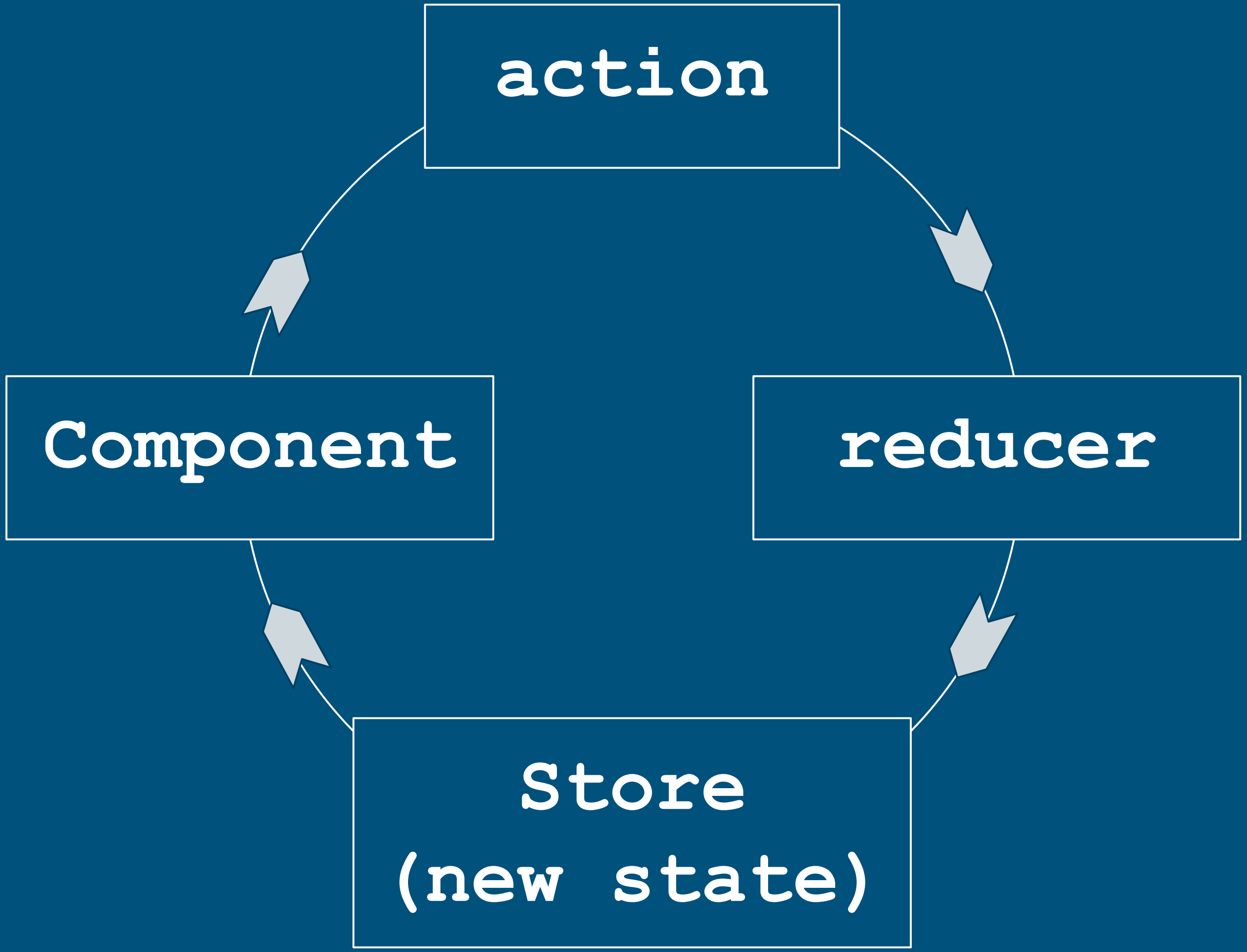
Similarly to the reduce() method, reducers take in data and reduce it to a single object: the state

Reducers are functions with a switch statement that returns a new state based on the action type sent.


action
Reducer (pure function)
New state
Current state

4. Unidirectional Flow

1. Component triggers an action
2. Action dispatched to reducer
3. Reducer returns the new state
4. Change in store causes rerender in components that rely on the piece of state that changed


action
reducer
Store
(new state)
Component

Why is Redux hard?

Action Creators

Reducers

Store

Components

mapStateToProps

combineReducers

ALL_THE_CAPS

Dispatch

Provider

connect

Separate Concerns

React-Redux	Redux	Convention
Provider	Reducers	ALL_THE_CAPS
connect	Store	Action Creators
Components	Dispatch	
mapStateToProps	combineReducers	

Redux Glossary

1. Redux
2. Store
3. Actions
4. Reducers
5. Dispatch