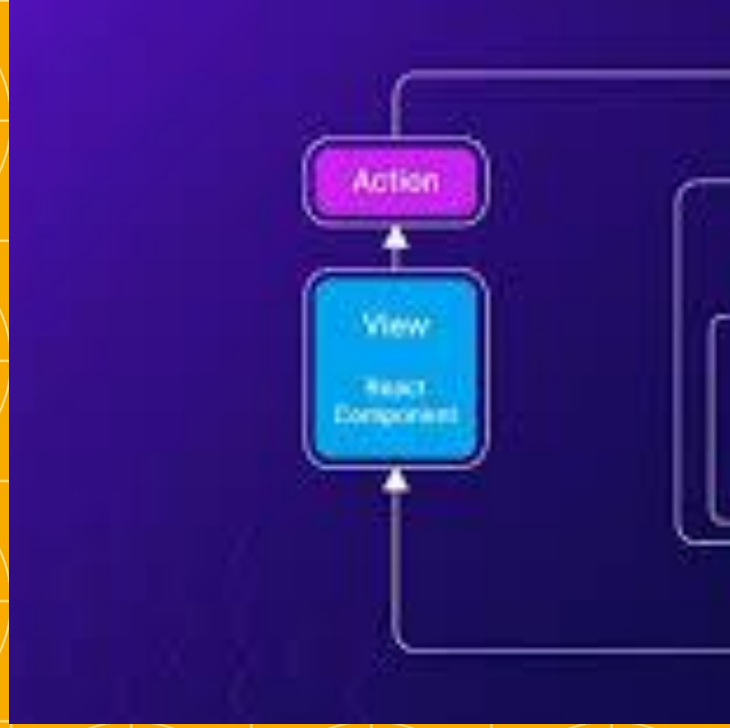
Google Developer Student Clubs
University of Toronto Mississauga

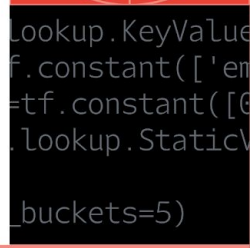
React & Redux Workshop: Building a Grade Tracking Application



Abdullah

Google Developer Student Clubs
University of Toronto Mississauga

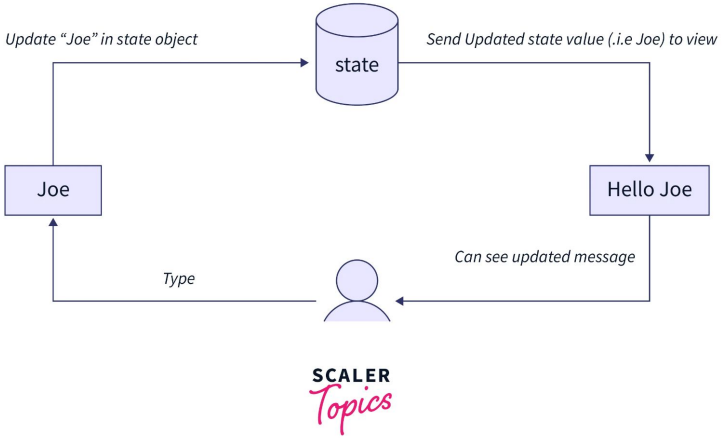
Prerequisites

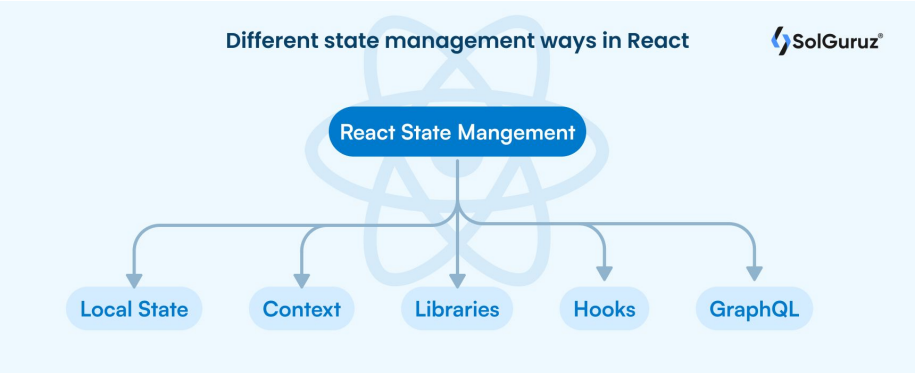


Learning objectives

- Learn about react state management!
- Learn what redux (and redux toolkit) are.
    - Store, actions, reducers
- Learn how to structure react apps when using redux
- Transform a static react app to a
- Persist redux store in localstorage (if time permits)

State


Update “Joe” in state object
state
Send Updated state value (.i.e Joe) to view
Joe
Hello Joe
Type
Can see updated message
SCALER
Topics


Different state management ways in React
SolGuruz
React State Mangement
Local State
Context
Libraries
Hooks
GraphQL

Prerequisites

- Basic javascript/typescript
- Basic react knowledge (components, hooks, using npm)
- Make sure your machine has npm installed.

Will not be going over how to install node and npm! You can google this if needed

State

- Local state
    - useState()
- Global state
    - useContext()
    - Redux
    - Zustand
    - And many more!

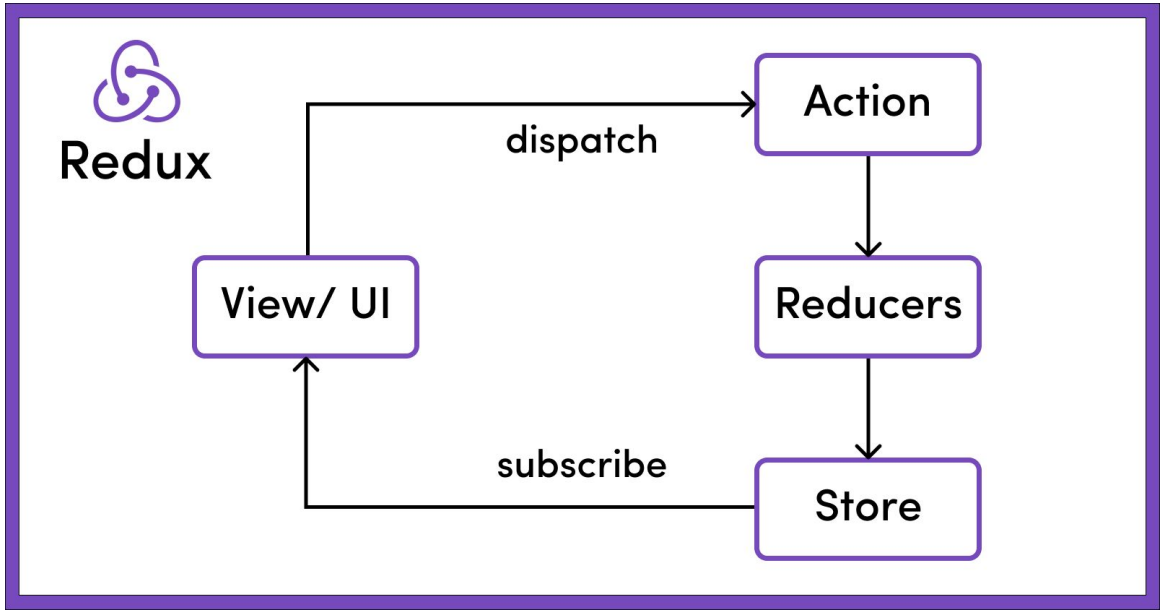
State

- What happens when a state variable updates?

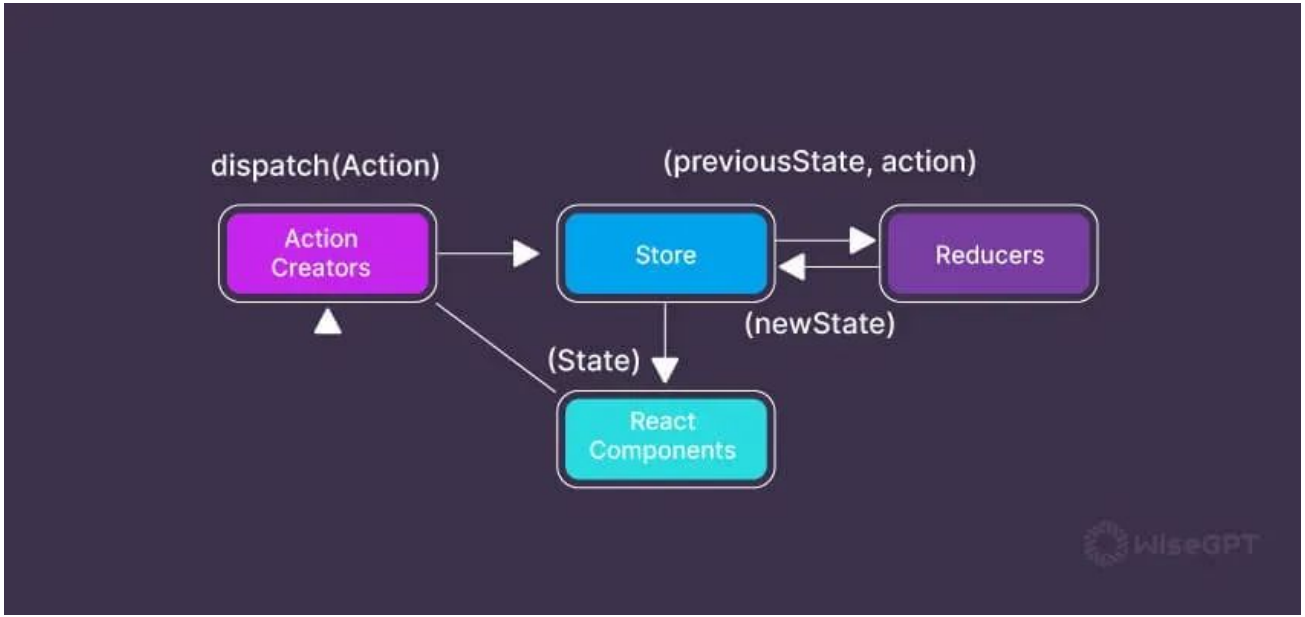
State

- What happens when a state variable updates?
    - Any component using that state is re-rendered.

Redux (General design pattern)

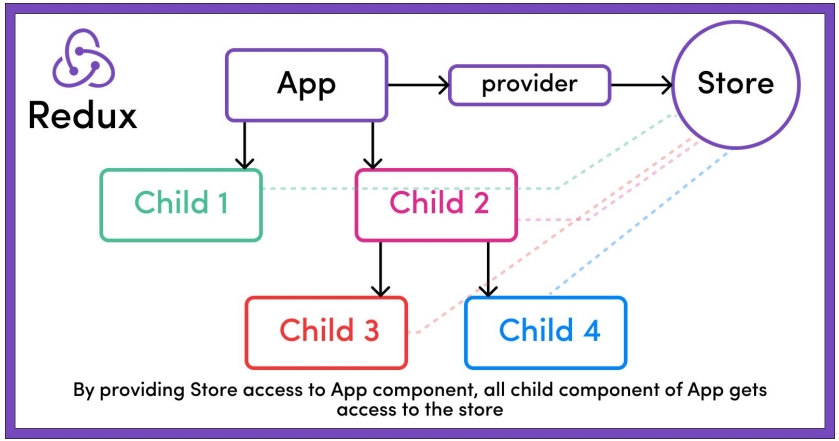


Redux (with React)



key concept: Redux store

The Redux store is the main, central bucket which stores all the states of an application. It should be considered and maintained as a **single source of truth** for the state of the application.



key concept: Actions

> *The only way to change the state is to emit an action, which is an object describing what happened – Redux Docs*

Core Concept: Actions are plain objects representing events that change the state, dispatched to the store to initiate updates.

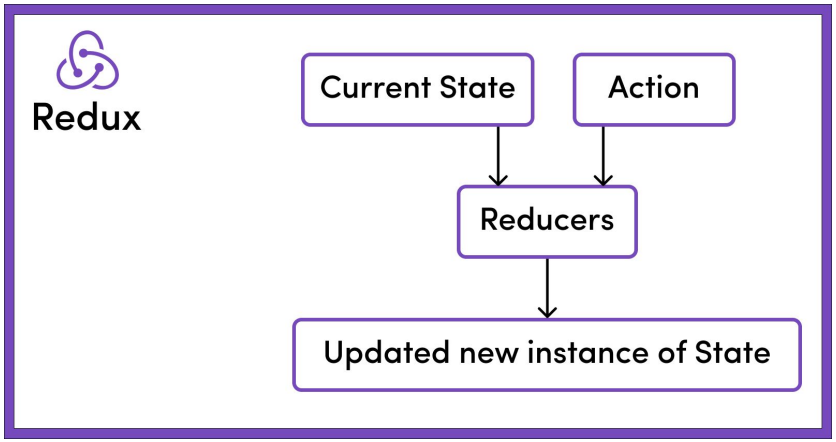
Note: Actions are the only way to interact with the store. if anyone wants to change the state of the application, then they'll need to express their intention of doing so by **dispatching an action**.

```
// Rest of the code

const dispatch = useDispatch()

const addItemToCart = () => {
return {
    type: "ADD_ITEM_TO_CART"
    payload: {
        bookName: "Harry Potter and the Goblet of Fire",
        noOfItem: 1,
        }
    }
}

<button onClick = {() => dispatch(addItemToCart())}>Add to cart</button>

// Rest of the code
```

key concept: Reducers

Reducers, as the name suggests, take in two things: **previous state** and **an action**. Then they reduce it (read it return) to one entity: the **new updated instance of state**.

Whenever an action is dispatched, **all the reducers are activated**. Each reducer filters out the action using a switch statement switching on the **action type**.

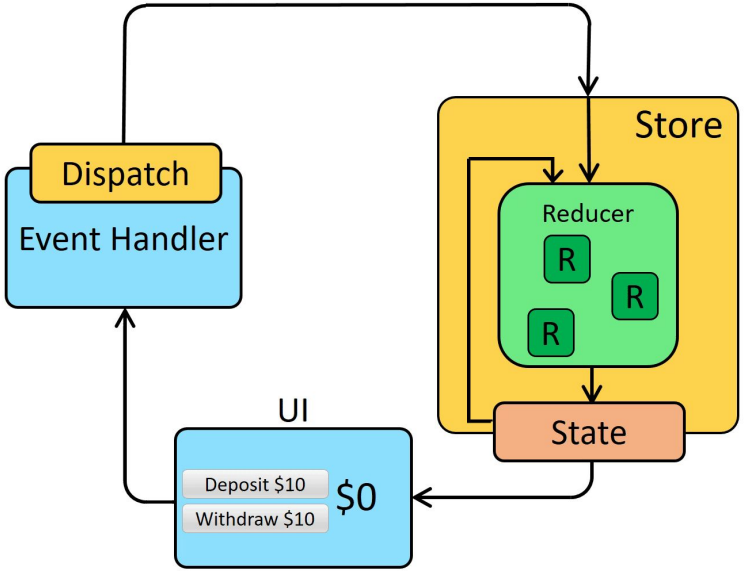


key concept: Selectors

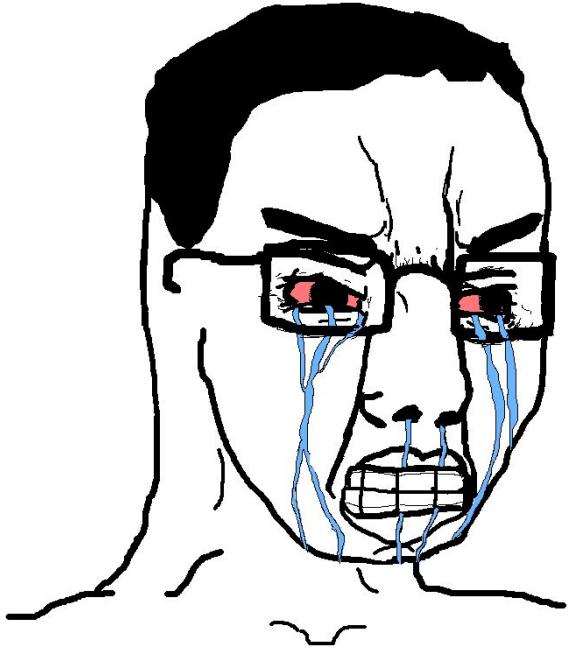
Core Concept: Selectors are functions used to retrieve data from the store, abstracting away the structure of the state.

Selectors make it easier to access nested state data, especially in large applications. They help keep components lean by encapsulating complex retrieval logic.

Putting it all together



Just useContext() bro



Redux vs Context

Why (and when) use redux instead?

- Designed for larger scale, enterprise applications
  - where state structure, immutability, and updates can get complicated. Redux Toolkit provides structure through slices, reducers, and actions, making it easier to manage and debug.
- Performance
  - State updates in context cause entire app to re-render!
  - Using selectors and only re-rendering components that rely on specific parts of the state reduces unnecessary re-renders.

Redux vs Context

Why (and when) use redux instead?

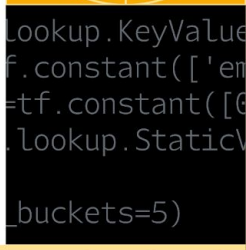
- Easier to debug!
  - RTK Integrates with powerful developer tools like Redux DevTools, which provides time-travel debugging, action history, and state snapshots, making it easier to understand and debug state changes.
- Scalability
  - Built to scale, with features like slices, standardized state management patterns, and modularity. It’s easier to add more slices and manage interactions between them as your app grows.

Average redux user

Google Developer Student Clubs
University of Toronto Mississauga

Lets build!

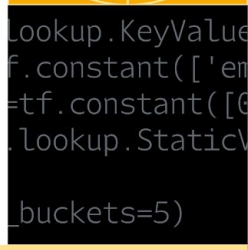
(https://github.com/nxabdullah/redux-gradetracker-workshop)



Google Developer Student Clubs
University of Toronto Mississauga

Set up the starter code (3 mins)

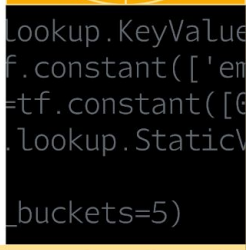
(https://github.com/nxabdullah/redux-gradetracker-workshop)



Google Developer Student Clubs
University of Toronto Mississauga

Codebase overview

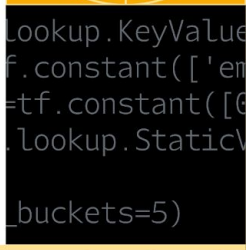
(https://github.com/nxabdullah/redux-gradetracker-workshop)



Google Developer Student Clubs
University of Toronto Mississauga

Creating grade and course slices

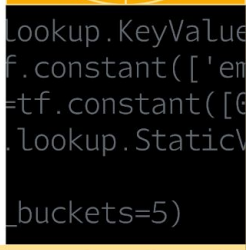
(https://github.com/nxabdullah/redux-gradetracker-workshop)



Google Developer Student Clubs
University of Toronto Mississauga

Course page!

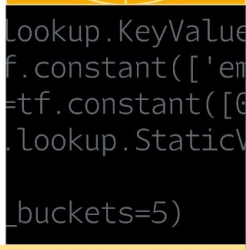
(https://github.com/nxabdullah/redux-gradetracker-workshop)



Google Developer Student Clubs
University of Toronto Mississauga

Grade page!

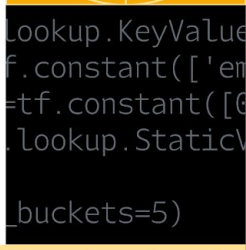
(https://github.com/nxabdullah/redux-gradetracker-workshop)



Google Developer Student Clubs
University of Toronto Mississauga

Persistence!

(https://github.com/nxabdullah/redux-gradetracker-workshop)



Google Developer Student Clubs

Thanks for attending!