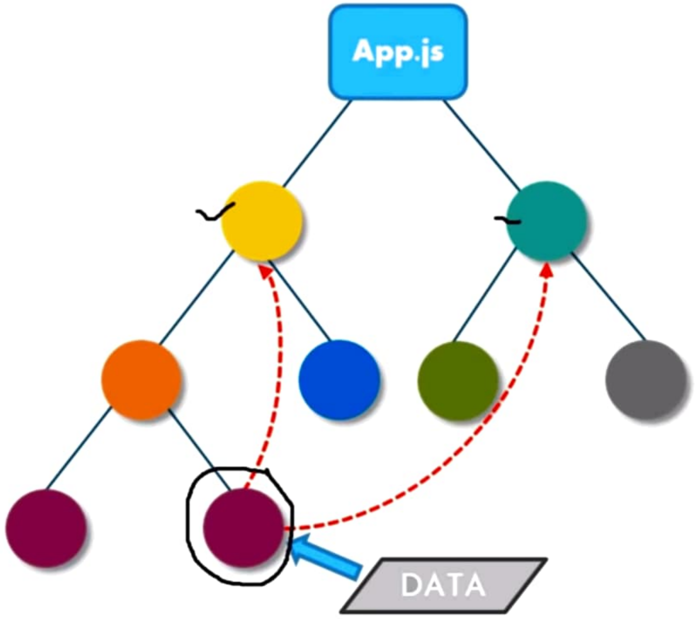
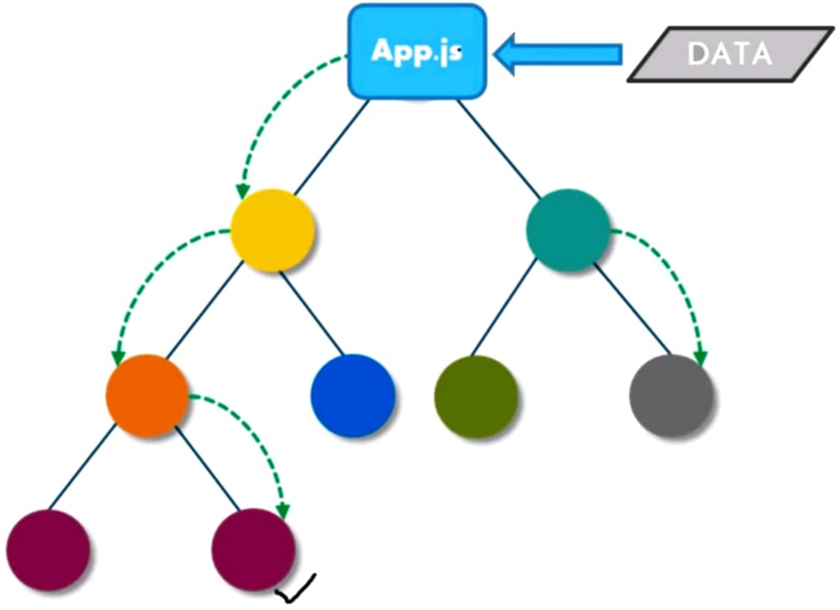
Why REDUX?

Normally in React How to pass data?

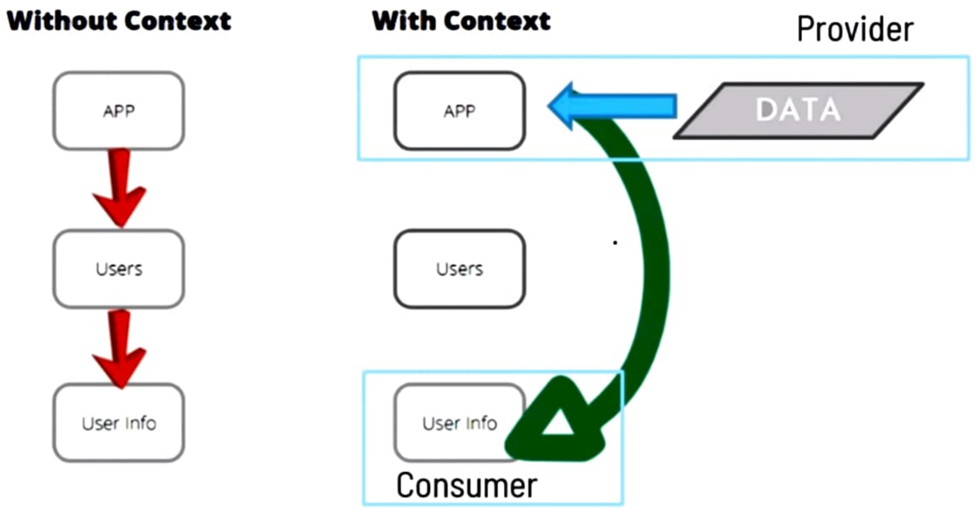
The data in React always flows from parent to child components which makes it unidirectional.



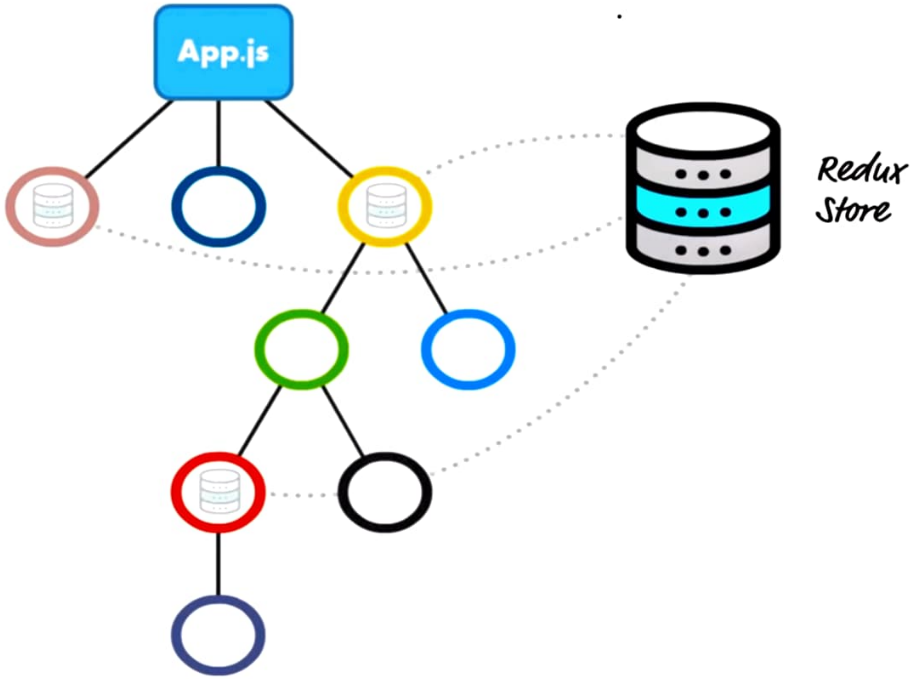
In React with Prop Drilling

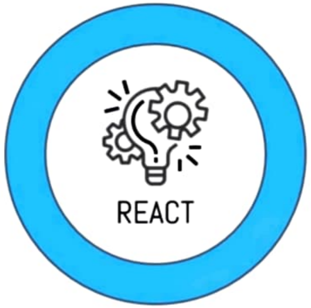


In React Hooks (context API, useContext)



How Redux Solve this Problem?





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.

REDUX Main Topics

01 ACTION

What to do?

02 REDUCER

How to do?

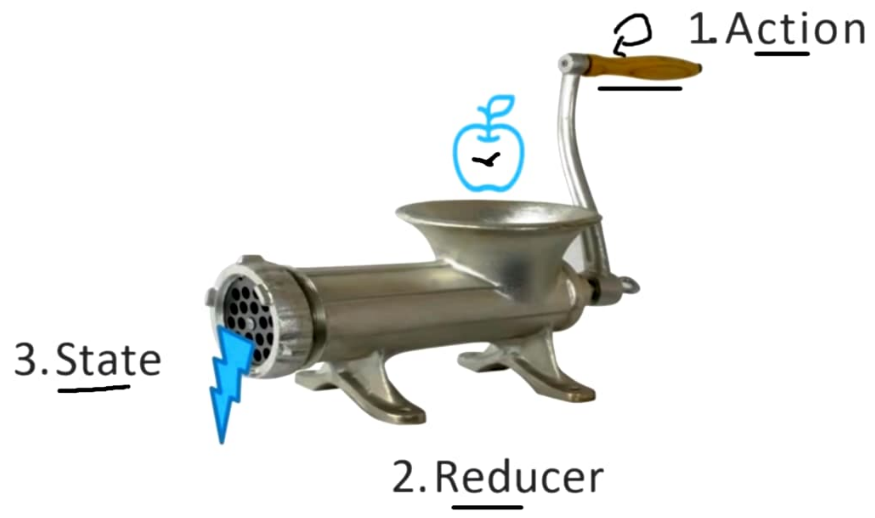
03 STORE

object which holds the state of the application

04 Functions associated with Store

createStore()
dispatch(action)
getState()

REDUX BASIC



1. Action

Pure **Object**

Actions are plain JavaScript **objects** that have a type field.
Actions only tell **what to do**, but they don't tell how to do.

```
return {
    type: 'INCREMENT',
    payload: num
  }
```

1. Action

Pure **Object**

```
return {
      type: 'DECREMENT',
      payload: num
  }
```

01 INCREMENT

When user click on Increment button.

02 DECREMENT

When user click on Decrement button.

1.1 Action Creator

Pure function which creates an action

```
export const incNumber = (num) => {
    return {
        type: 'INCREMENT',
        payload: num
    }
}
```

Reusable, Portable, and Easy to Test

02

Reducer

2. Reducer

Reducers are functions that take the current state and an action as arguments, and return a new state result.

```
const initialState = 0;

const changeTheNumber = (state = initialState, action) => {
   switch (action.type) {
     case "INCREMENT": return state + action.payload;
     case "DECREMENT": return state - 1;
     default: return state;
  }
}
```

03
Store

3. Store

The Redux store brings together the state, actions, and reducers that make up your app.

It's important to note that you'll only have a single store in a Redux application.

Every Redux store has a single root reducer function.

```
import {createStore} from “redux”;

const store = createStore(rootReducers);
```

REDUX Principles

01 **Single source of Truth**

The global state of your application is stored as an object inside a **single store**.

02 **State is Read-Only**

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

03 **Immutability, One-way data flow, Predictability of outcome**

04 **Changes are Made with Pure Reducer Functions**