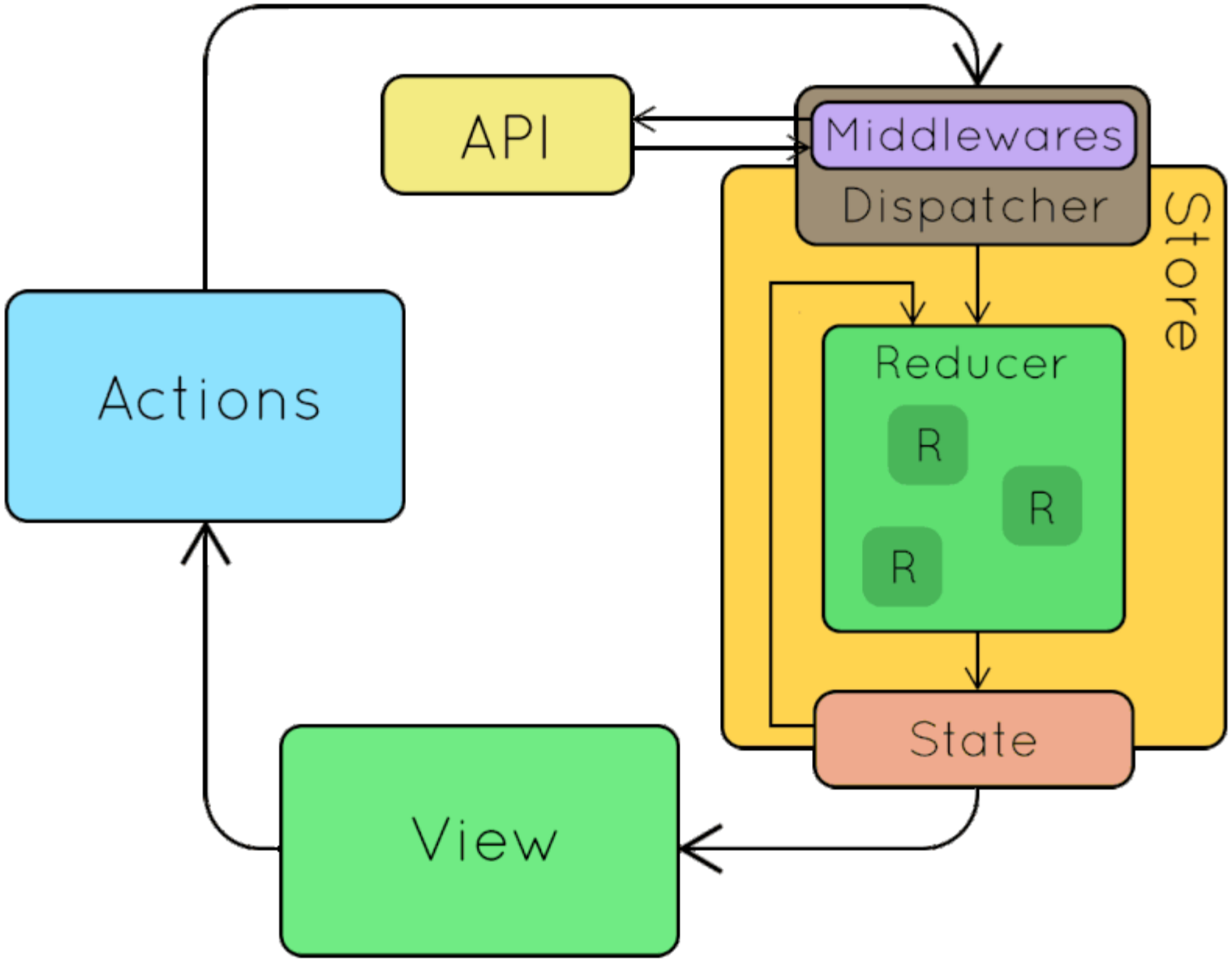
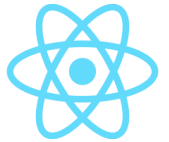
REDUX

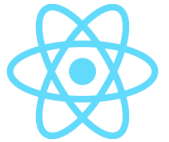

API
Middlewares
Dispatcher
Store
Actions
Reducer
R
R
R
State
View



intro useReducer() hook

https://react.dev/reference/react/useReducer

implement Redux in js

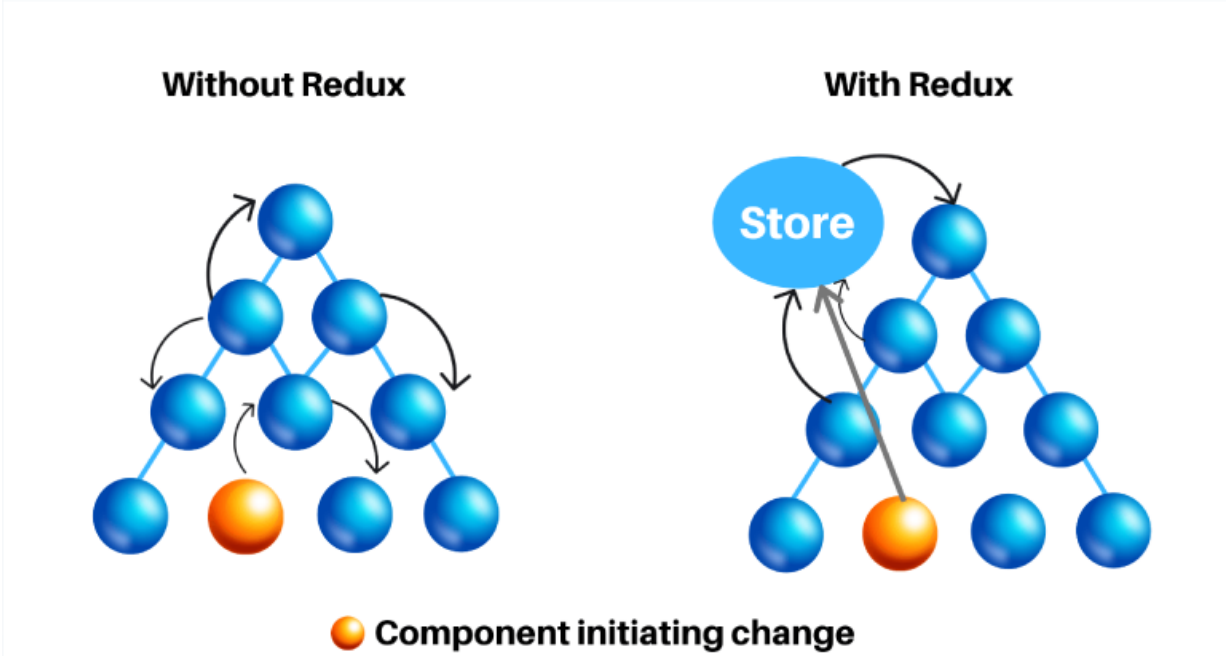


What is Redux?

Redux is a predictable state container for JavaScript apps.

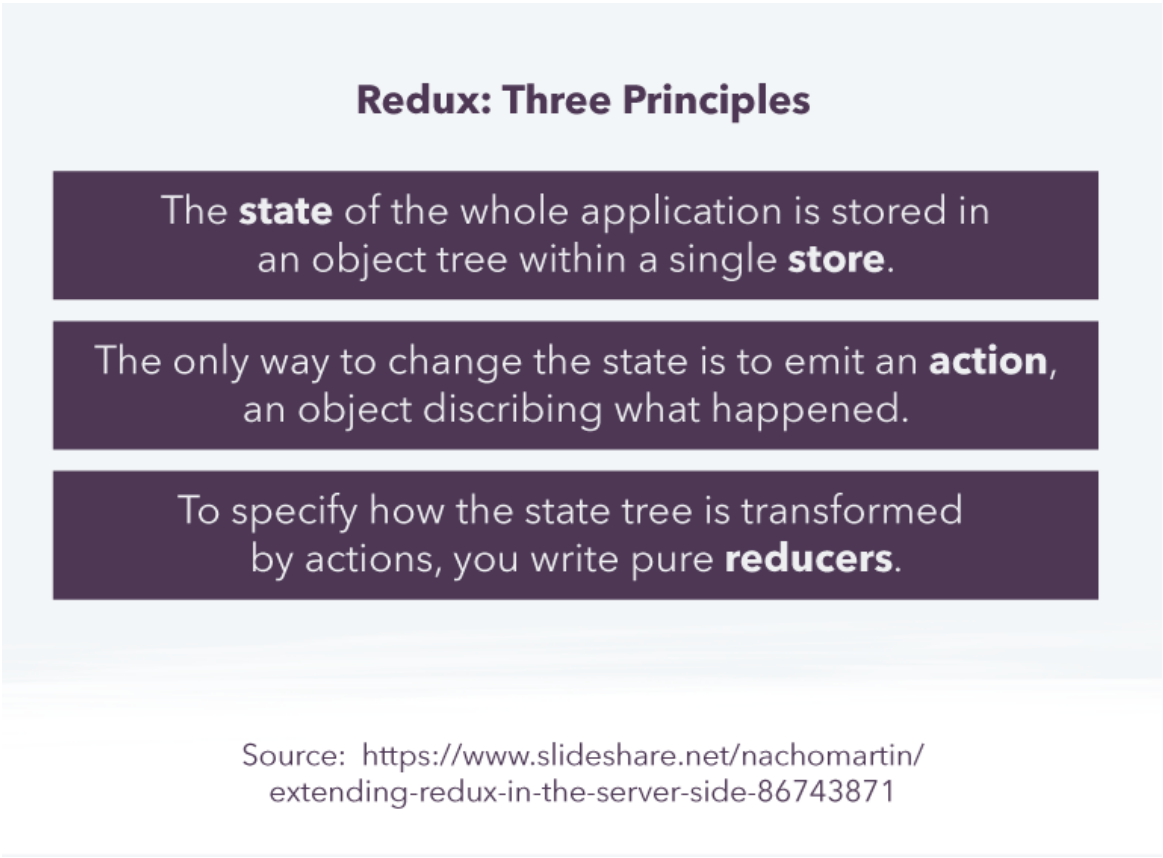
Redux allows you to manage your app's state in a single place and keep changes in your app more predictable and traceable, making it easier to understand the changes happening in your app

Why Redux?



Source: https://synergytop.com/blog/the-advantages-of-using-redux-along-with-react/

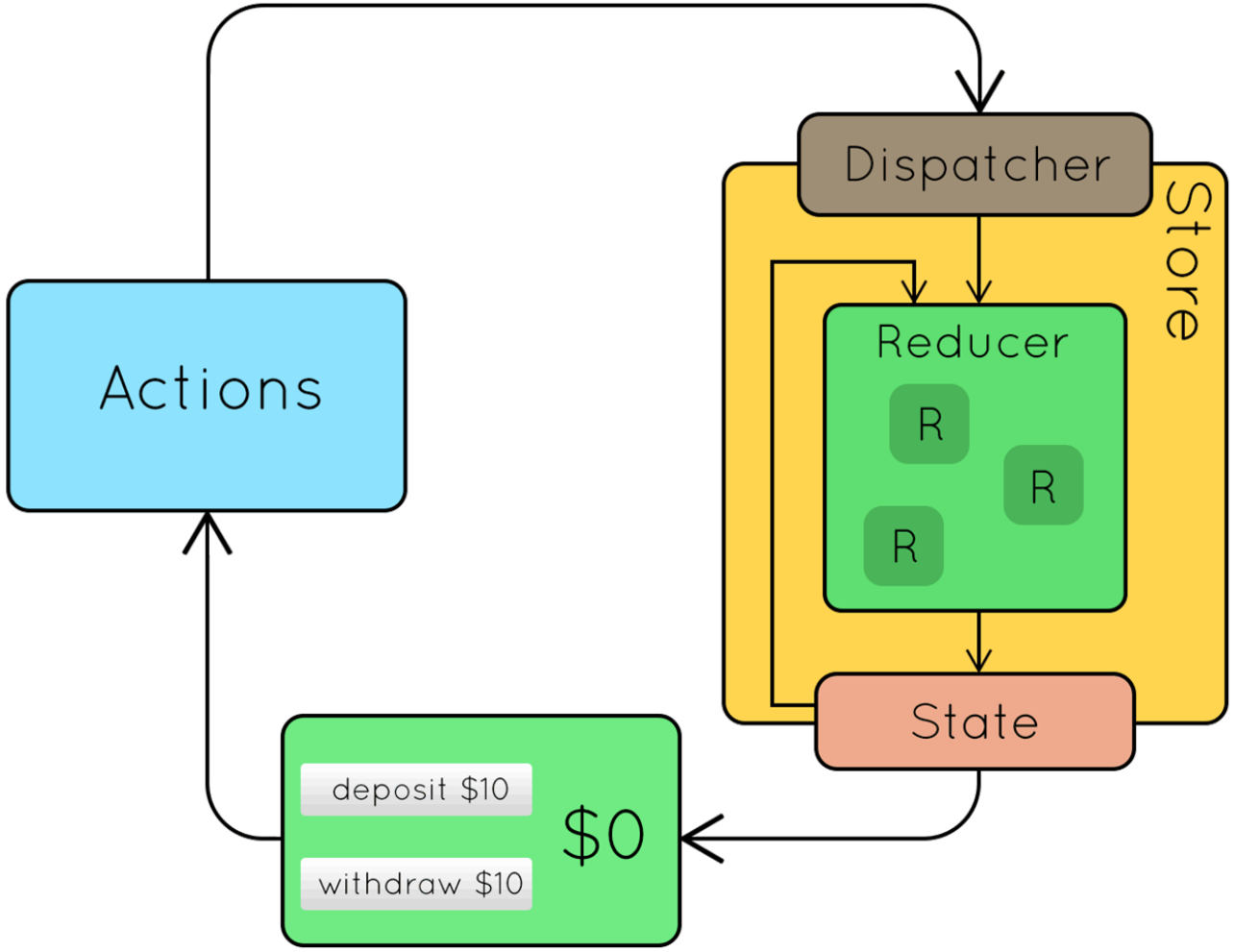
Why Redux?


Redux: Three Principles
The state of the whole application is stored in an object tree within a single store.
The only way to change the state is to emit an action, an object discribing what happened.
To specify how the state tree is transformed by actions, you write pure reducers.
Source: https://www.slideshare.net/nachomartin/extending-redux-in-the-server-side-86743871

When you should use Redux?

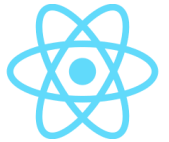
- **When multiple components need to access the same application state.** For example, once a user logs in to a personal account on an application, the information about that user needs to be shared with a range of components that may not interact directly. In this case, Redux's single store is an optimized way to map a state across components.
- **When you're working on a large application with several other people.** Again, the Redux single store becomes more beneficial as the complexity of an application increases. If many people are working on it (and you don't have an alternative system for sharing information), then Redux can also help keep everyone on the same page.
- **When application state is updated frequently.** This often implies consumer-facing applications, where there is a lot of user interaction (for example, adding goods to a basket, then removing some, going through the payment process, etc).

Redux Core


Actions
Dispatcher
Store
Reducer
R
R
R
State
deposit $10
withdraw $10
$0

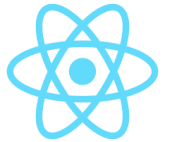
Redux Core

```
npm install redux react-redux --save
```



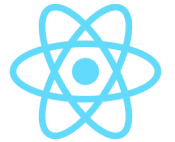
Redux Core

- Actions: **events**
- Reducers: **pure function**
- Store: **object**



Action Code Example

- Actions: **events**
- Reducers: **pure function**
- Store: **object**
- Dispatch: **function**

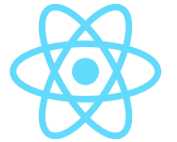


Action Example: object or function return object

```
export const increment = () => ({
  type: "INCREMENT",
});
export const decrement = () => ({
  type: "DECREMENT",
});
```

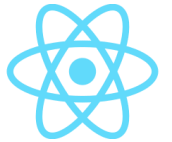
Action Example: object or function return object

```
const incrementCreator = (data)=>{
    return {
        type:'increment',
        payload:data
    }
}
```



Reducer Example: function(s) switch case

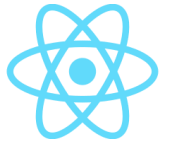
```
export const incrementReducer = (state = initialState, action) => {
  switch (action.type) {
    case "INCREMENT":
      return {
        ...state,
        count: state.increment.count + 1,
      };

    default:
      return state;
  }
};
```



store

createStore(reducer, initValue, enhencers)

```
import {createStore} from 'redux'
import myReducer from './reducers';

💡
const store = createStore(myReducer);

export default store;
```

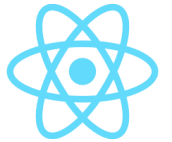


dispatch()

- dispatch(action)

combine reducers

```
import { combineReducers } from "redux"
import counterReducer from "./counterReducer"


const myReducer = combineReducers({
    counter: counterReducer
})

export default myReducer;
```



connect function

```
export default connect(mapStateToProps, mapDispatchToProps)(Counter);
```

mapStateToProps function

```
const mapStateToProps = (state) => {
  console.log(state);
  return  ({count: state.counter.count,})
};

const mapDispatchToProps = {
  increment,
  decrement,
};
```

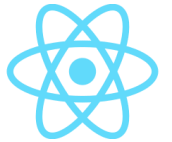
more

```
import { StyleSheet, Text, View, Pressable } from "react-native";
import React from "react";

import { connect } from "react-redux";

import { increment, decrement } from "./store/actions/counterAction";

const Counter = ({count, increment, decrement}) => {
  return (
    <View style={{ flex: 1, alignContent: "center", alignItems: "center" }}>
      <Text style={{ margin: 30 }}>App {count}</Text>
```

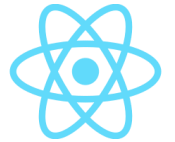
Provider

```
const App = () => {
  return (
    <Provider store={store}>
      <Counter/>
    </Provider>
  );
};
```

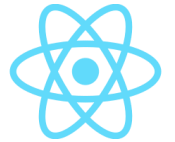
Example-Counter-Todo App



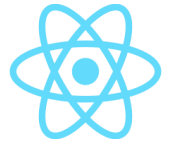
Redux Thunk



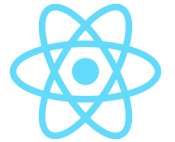
Redux Saga



Redux DevTool



Redux Toolkit




THANK YOU!