Google | 1000 طويق

# Redux

أكاديمية طويق
TUWAIQ ACADEMY

وزارة الاتصالات وتقنية المعلومات
MINISTRY OF COMMUNICATIONS AND INFORMATION TECHNOLOGY

الاتحاد السعودي للأمن السيبراني والبرمجة والدرونز
SAUDI FEDERATION FOR CYBERSECURITY, PROGRAMMING & DRONES

مهارات المستقبل

stc

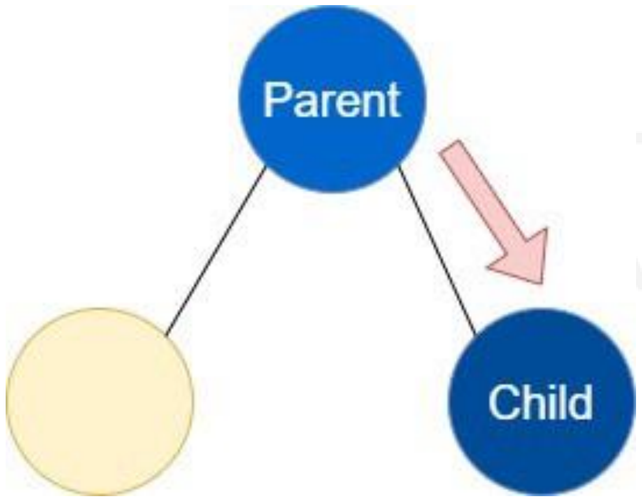
# Objective

- Component communication
- What is Redux?
- Why to use Redux?
- Redux flow

# Component communication

- React allows component communication in the following ways:

1. Parent to Child:




```
{ Component } from 'react'

 class ParentComponentProps extends Component {
r() {
);
tate = {name: 'Mark'};


 (
iv>
  <ChildComponent name="Sumeet" /> {/* For Static Value */}
  {/* Or */}
  <ChildComponent name={this.state.name} /> {/* For Dynamic Va
div>




ponent extends Component {

 (
1>Hi {this.props.name}</h1>
```

   - Using Props:

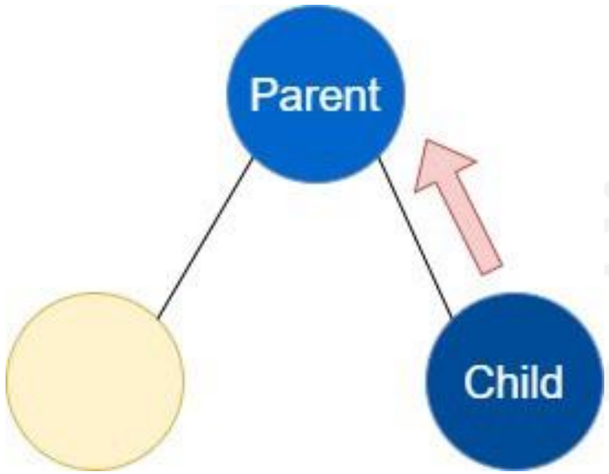
# Component communication

- Using the Instance method

```
import React, { Component } from 'react'
export default class UsingInstanceMethod extends Component {
    componentDidMount() {
        this.foo.changeName('Mark');
    }
    render() {
        return (
            <div>
                <ChildComponent ref={ref => this.foo = ref}/>
            </div>
        )
    }
}
class ChildComponent extends Component {
    constructor() {
        super();
        this.state = {name: ''};
    }
    changeName(name) {
        this.setState({name});
    }
    render() {
        return (
            <h1>Hi {this.state.name}</h1>
        );
    }
}
```

# Component communication

2. Child to Parent




1. Create a callback method in parent and pass it to the child using props.

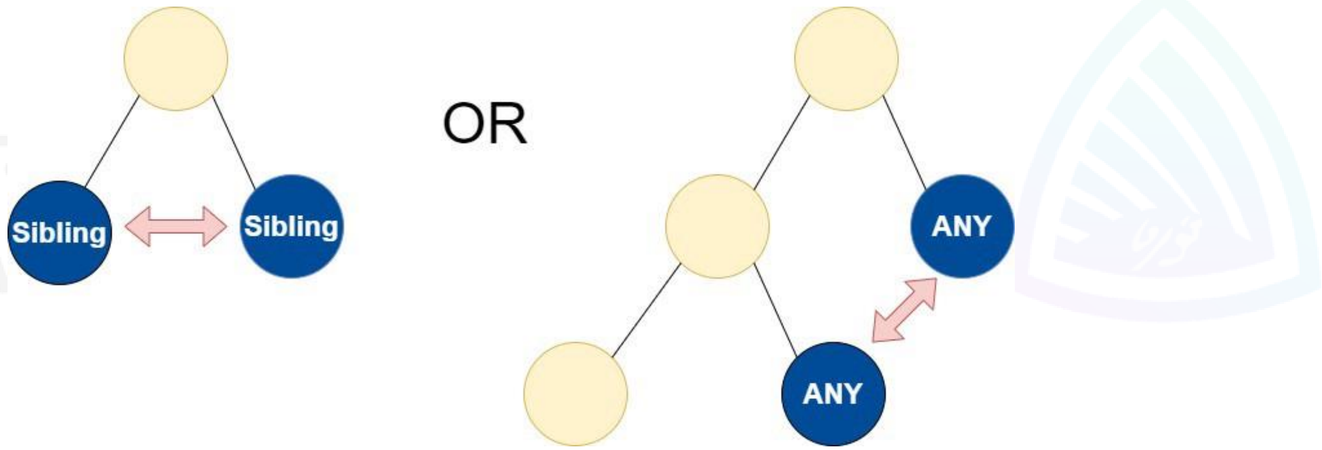
2. Child can call this method using "**this.props.[yourCallbackName]**" form child and pass data as argument.

```
export default class UsingCallback extends Component {
    constructor() {
        super();
        this.state = {name: 'Sumeet'};
    }
    setName = (name) => {
        this.setState({name});
    }
    render() {
        return (
            <div>
                <h1>Hi {this.state.name}!</h1>
                <ChildComponent setName={this.setName}/>
            </div>
        )
    }
}
class ChildComponent extends Component {
    sendParent = () => {
        this.props.setName('John');
    }
    render() {
        return(
            <button onClick={this.sendParent}>Click Me</button>
        )
    }
}
```

Google | طويق

# Component communication

3. Communication between Sibling / Any components:




- React support one-way data flow. In order to send data from **sibling 1** to **sibling 2,** you have to send data to **parent** and then from **parent** to **sibling 2**. This approach is a little complicated and difficult to maintain.

# What is Redux?

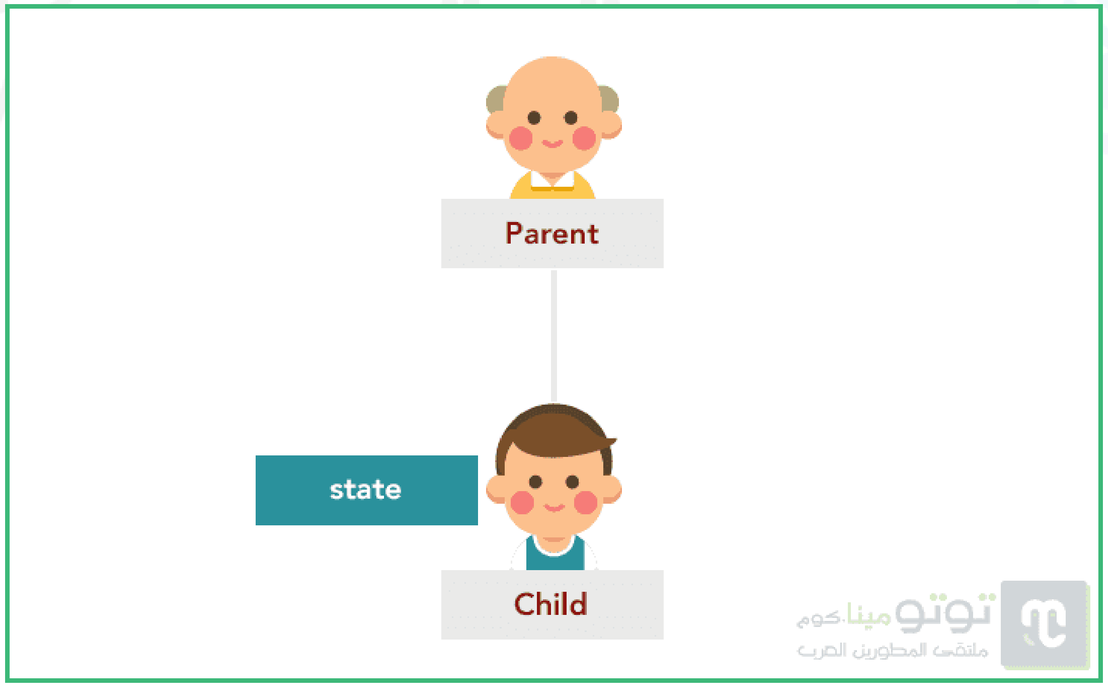
- State container that can be used with React to assist with site management
- It stores all state information for an app in one central place.
- The state information is stored in the store and the entire state of the application is stored in a single object known as the state tree.

Redux

# Why to use Redux?

- We may be able to create complete applications using React.js, but we will soon discover the limitations of our options, especially when the complexity of the application increases and the interrelationships between its various components (components) increase.

- For example, suppose we have a father component that has one son, and the latter has its own state. Note the following picture:

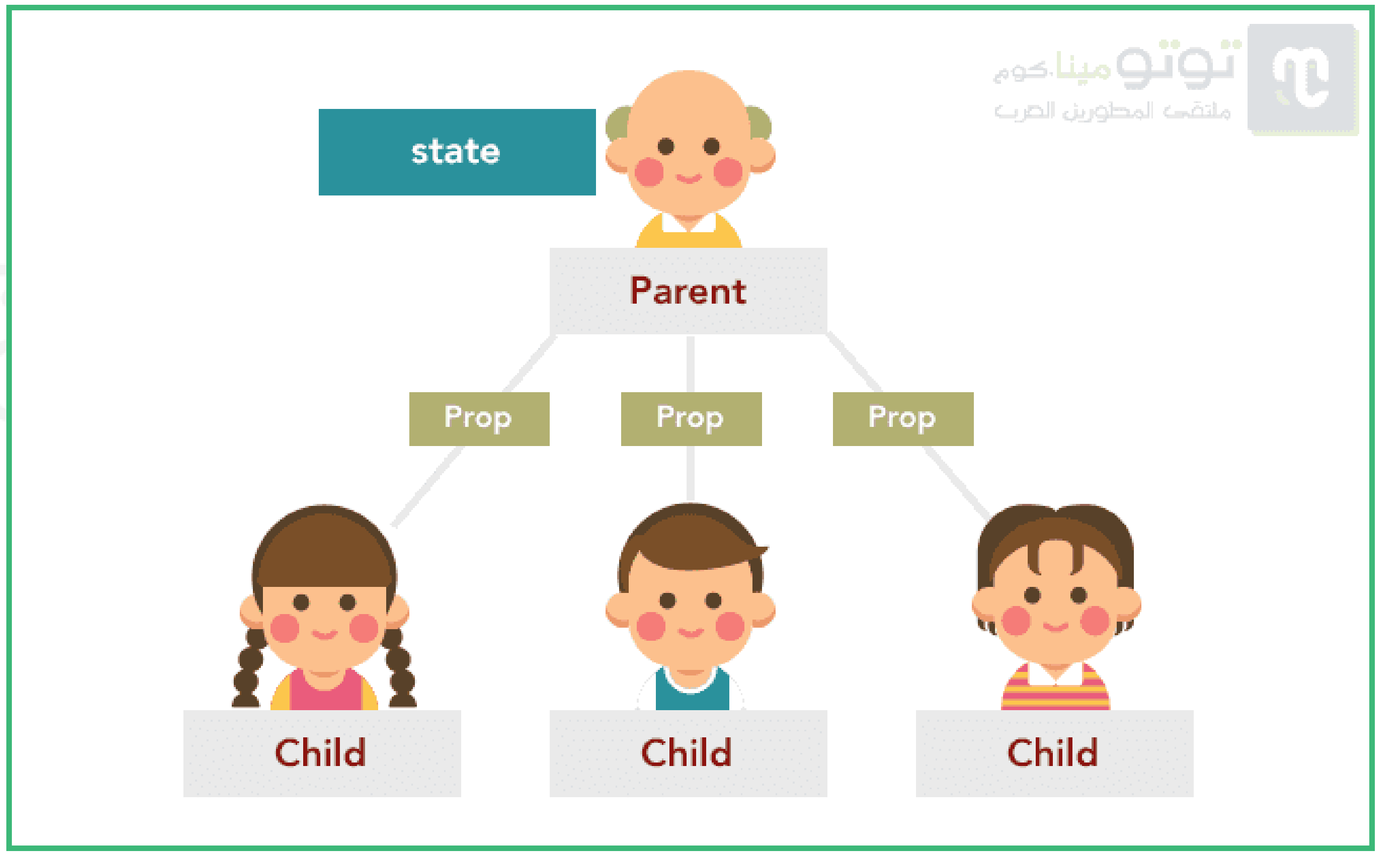


# Why to use Redux?

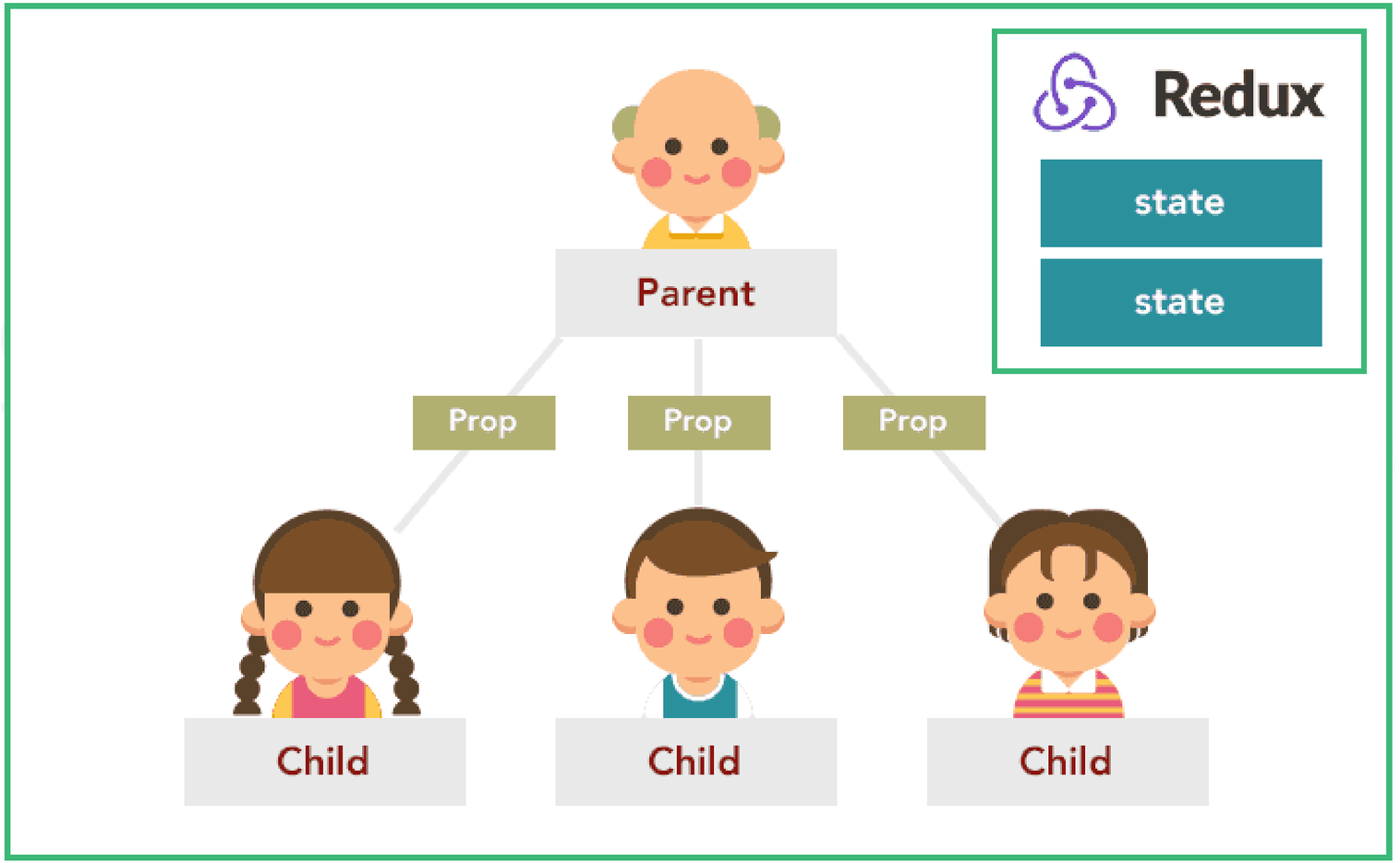
- When we started working on our project we didn't have a complete idea of what the final application tree would look like (and it's really hard to tell), we thought that the parent component in the image would have an only son so we gave the latter a state of its own.

- After a while, we realized that the father will have other children who need to access the state of the first child, which is impossible in React.js because the state is passed from top to bottom only by Props, and cannot be passed horizontally between components of the same level.

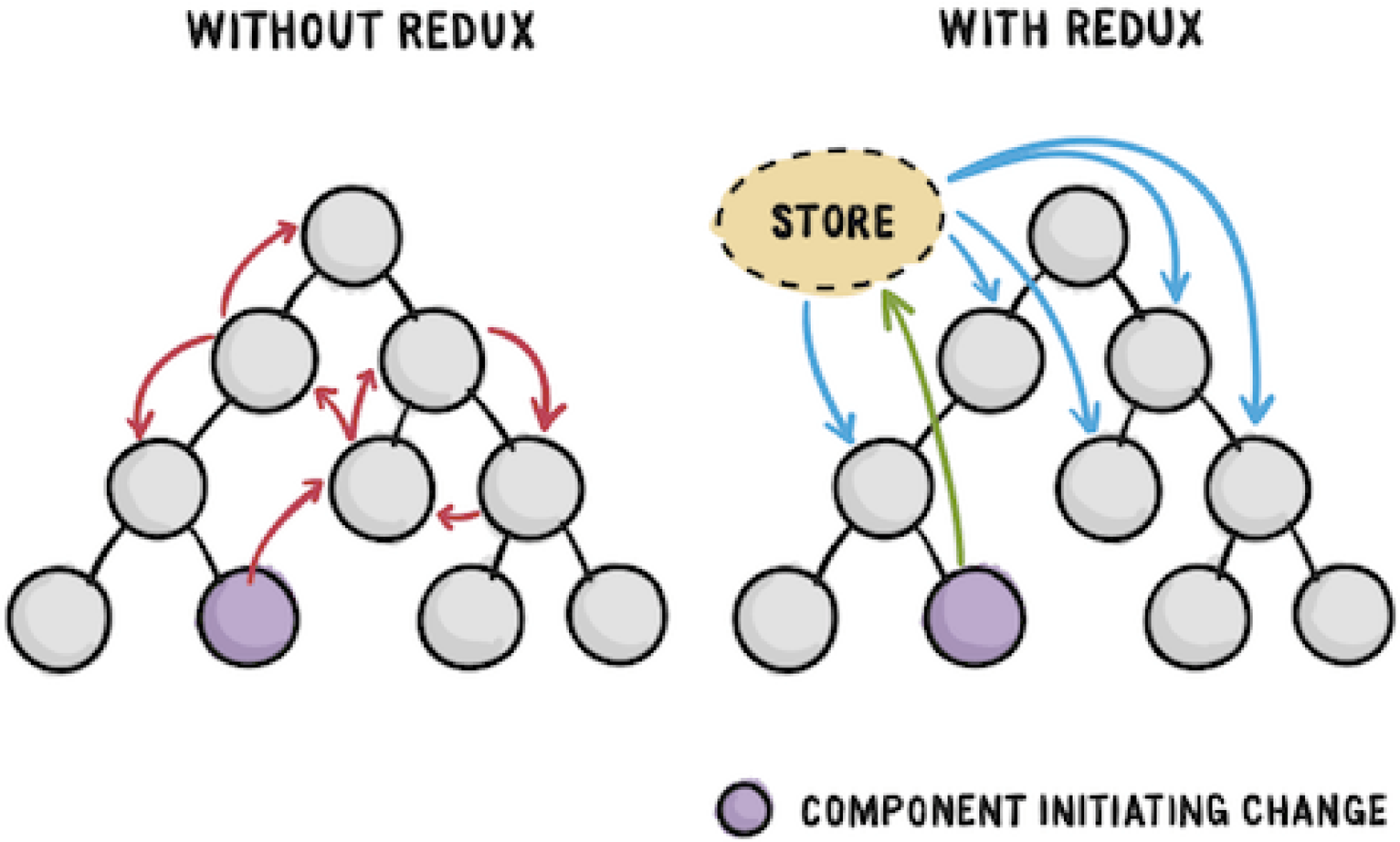
- So the only solution will be to strip the first son of his state and raise it to the father so that the father can pass it again to the three sons in the form of Prop.



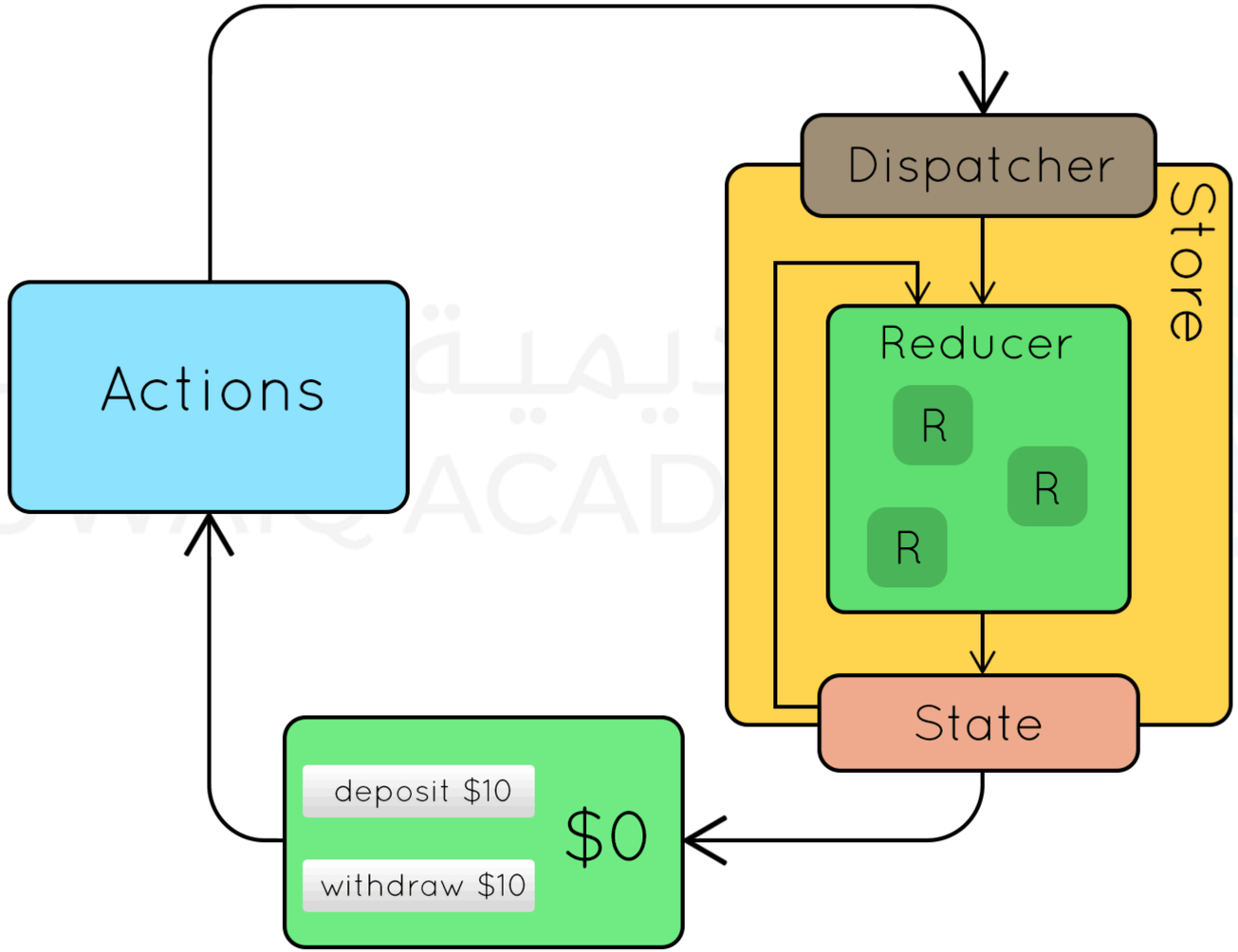
# The Solution is: using Redux



# The Solution is: using Redux

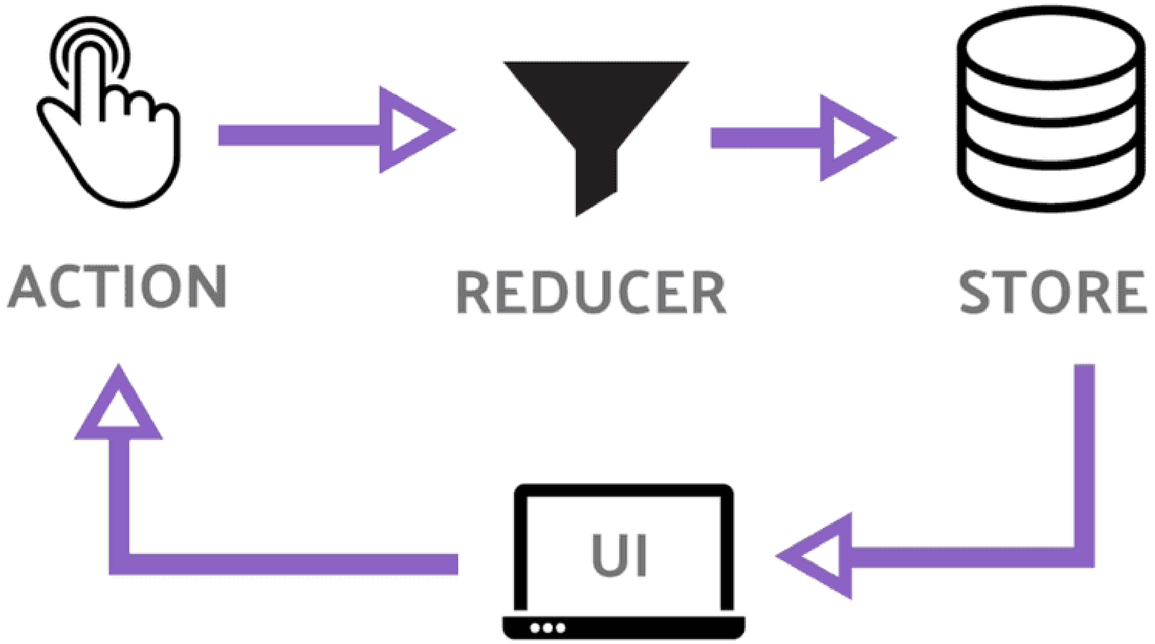


# Redux flow


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

# Redux flow

- It all starts from the User Interface, where the user performs a specific behavior that translates in Redux into what is known as Action.
- This action is received in pure functions known as Reducers. It usually accepts two arguments: the old state and the previously fired action, and returns the new state based on the information in the action.
- The action is an object with two basic properties:
  - type : This property is mandatory and contains the type of action to be performed.
  - payload : This property is not mandatory - it can be called any name other than payload - and it contains the information that we want to send to the Store to be merged into the state of the application via Reducers.



# Resources

- https://www.tutomena.com/web-development/javascript/redux-library/
- https://giters.com/Lazytangent/thunks
- https://medium.com/@sumeet.ru/component-communication-in-react-without-redux-5006b7a6009d