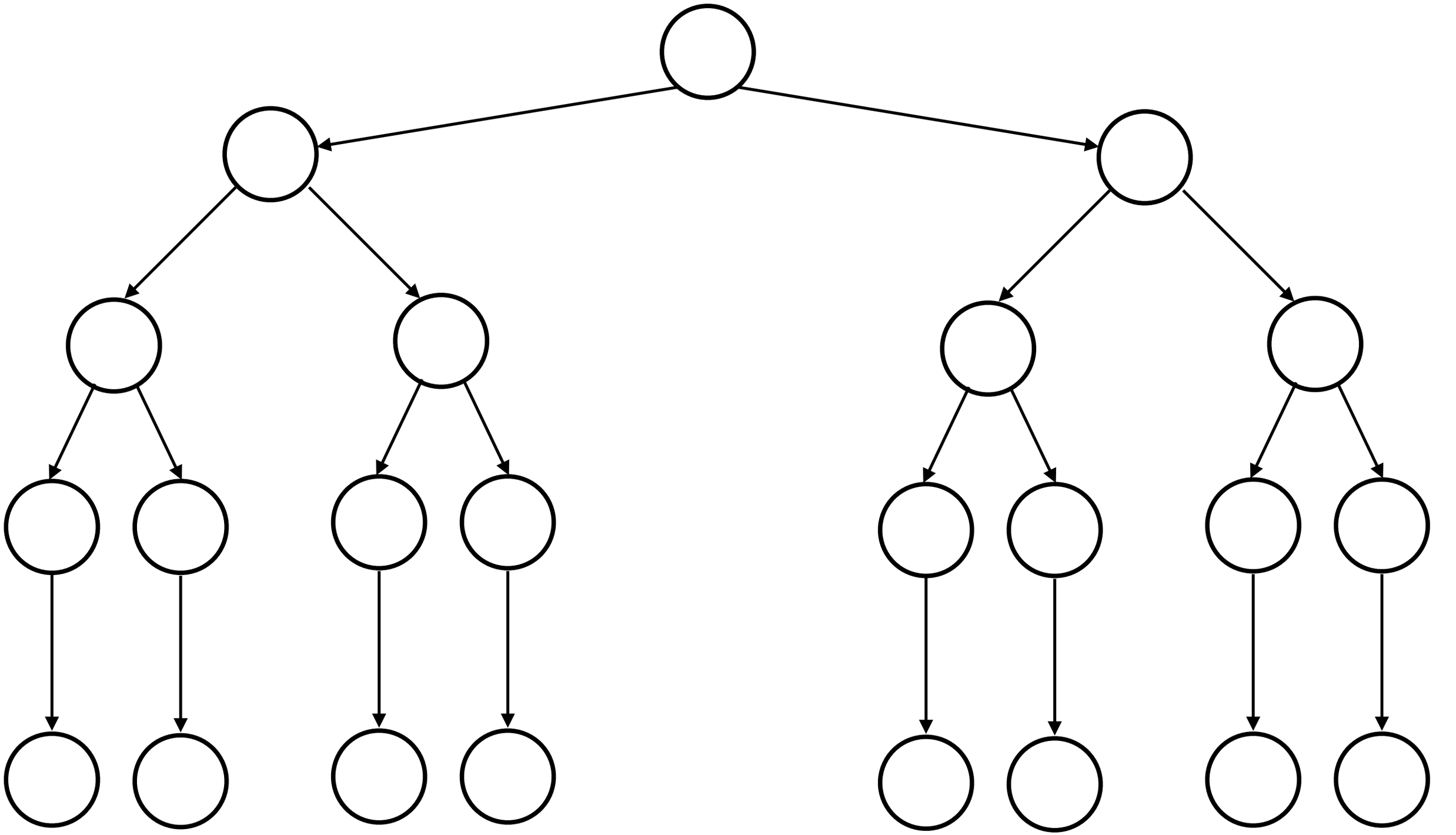
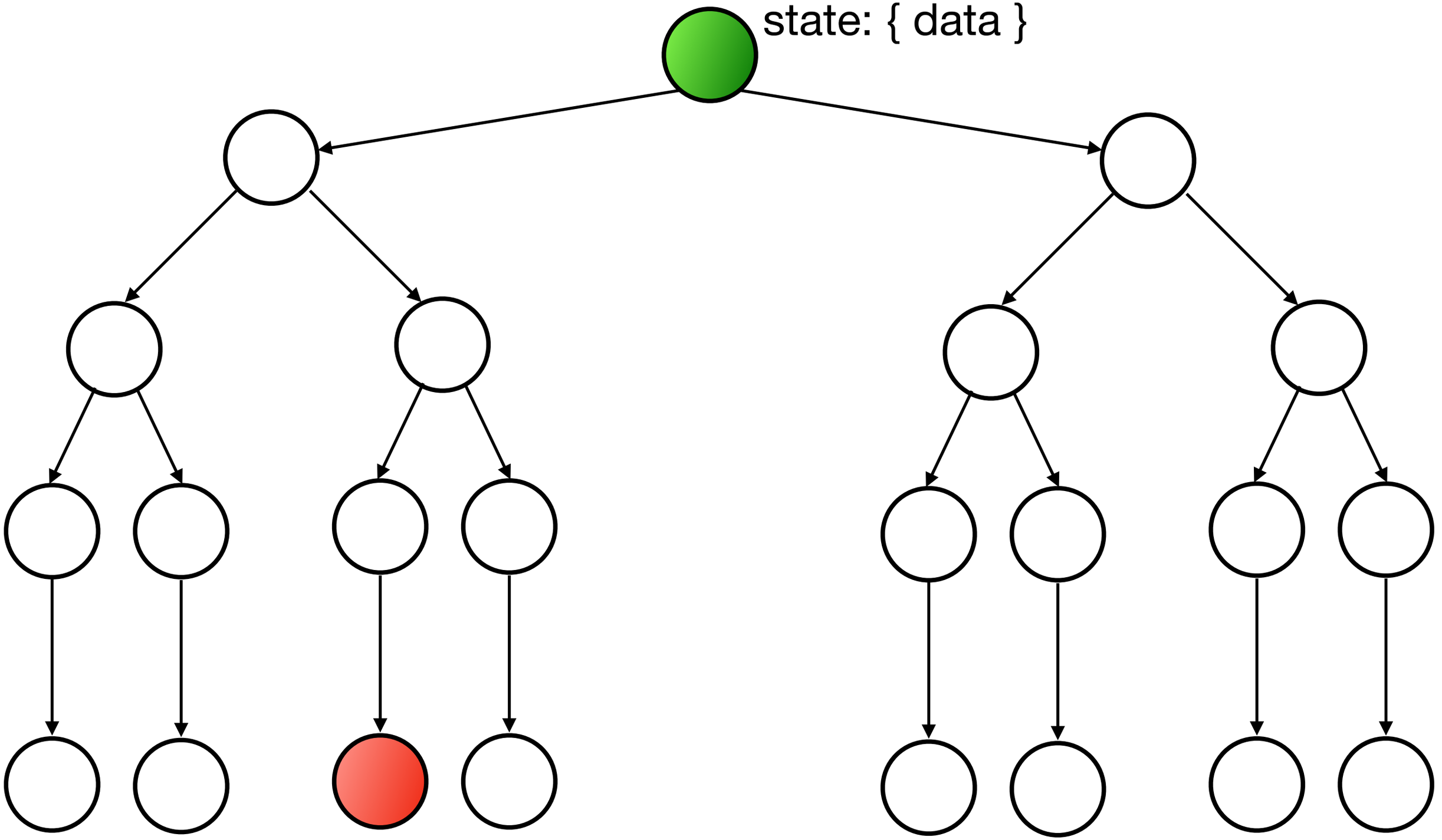
Redux

Необходимые модули: redux, react-redux

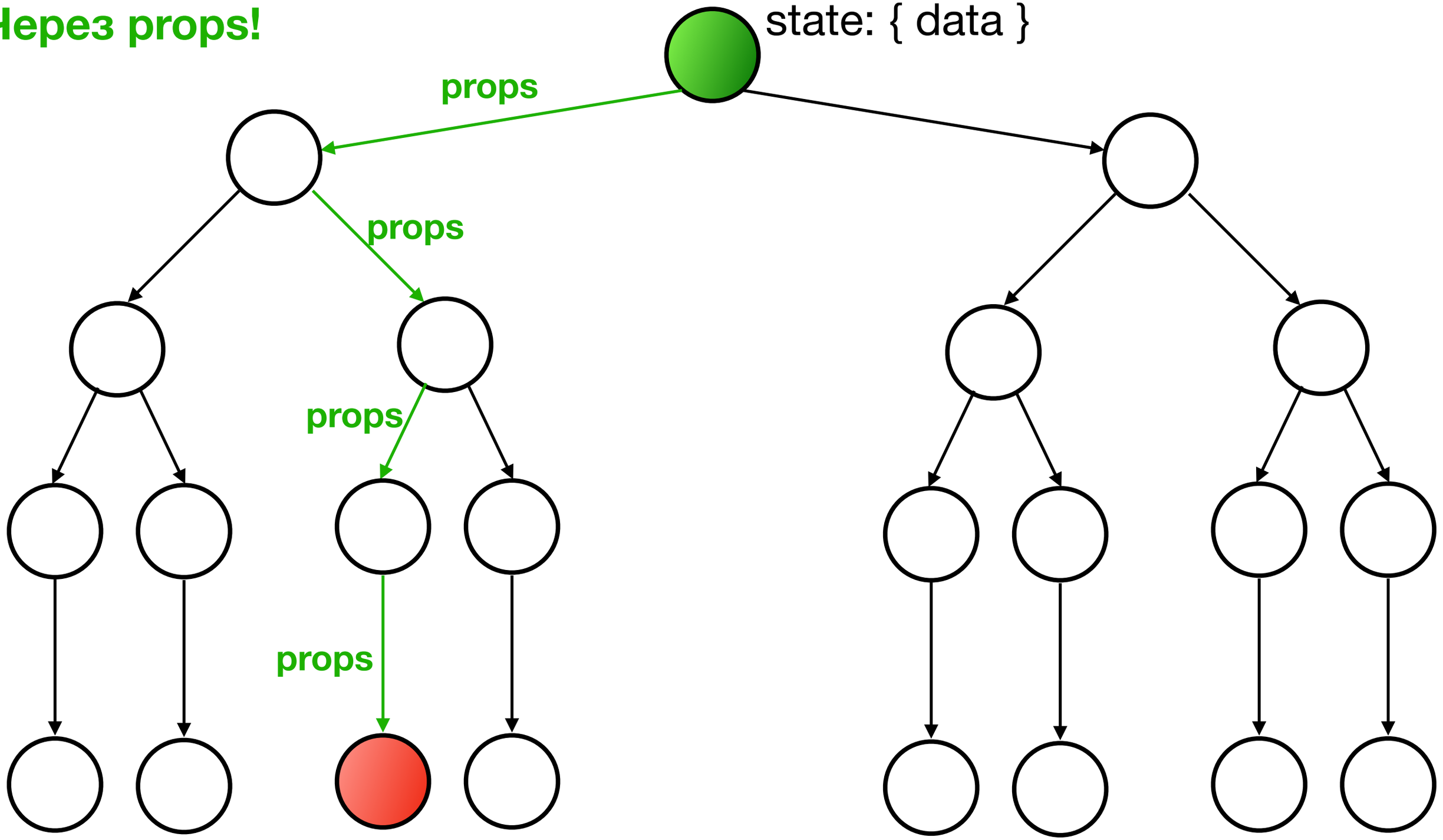


Как красному компоненту получить данные из зеленого?

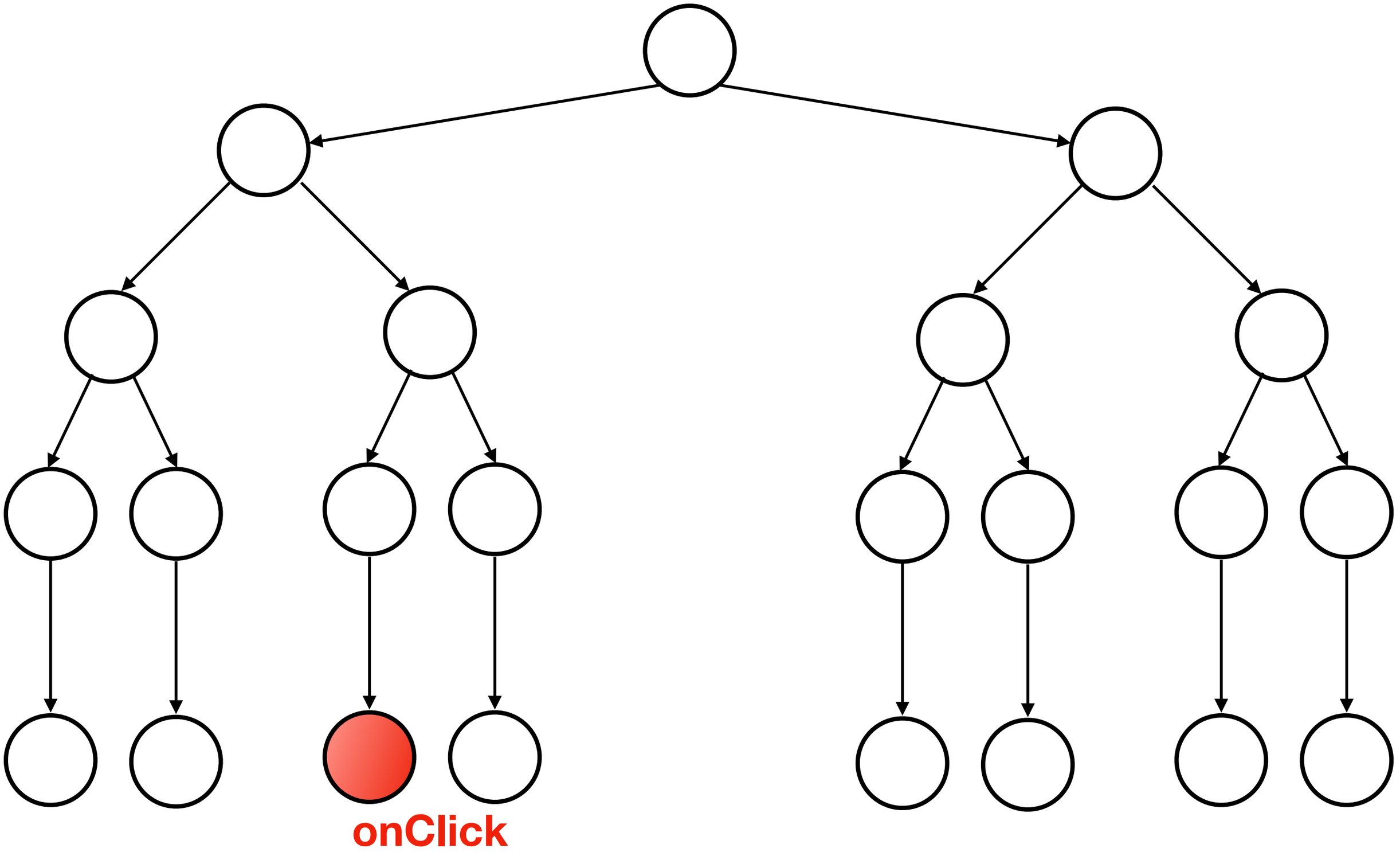


Как красному компоненту получить данные из зеленого?

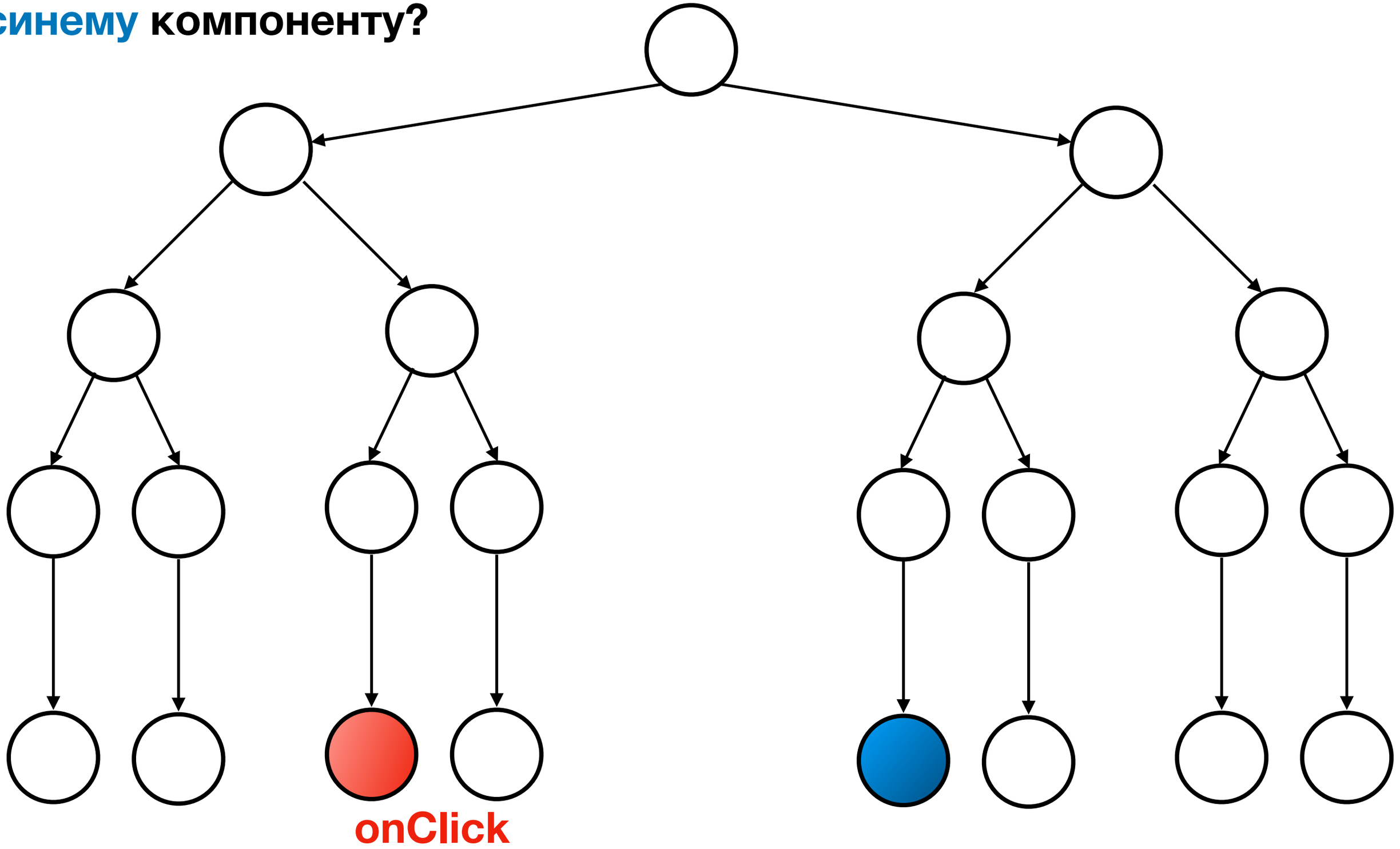
Через props!



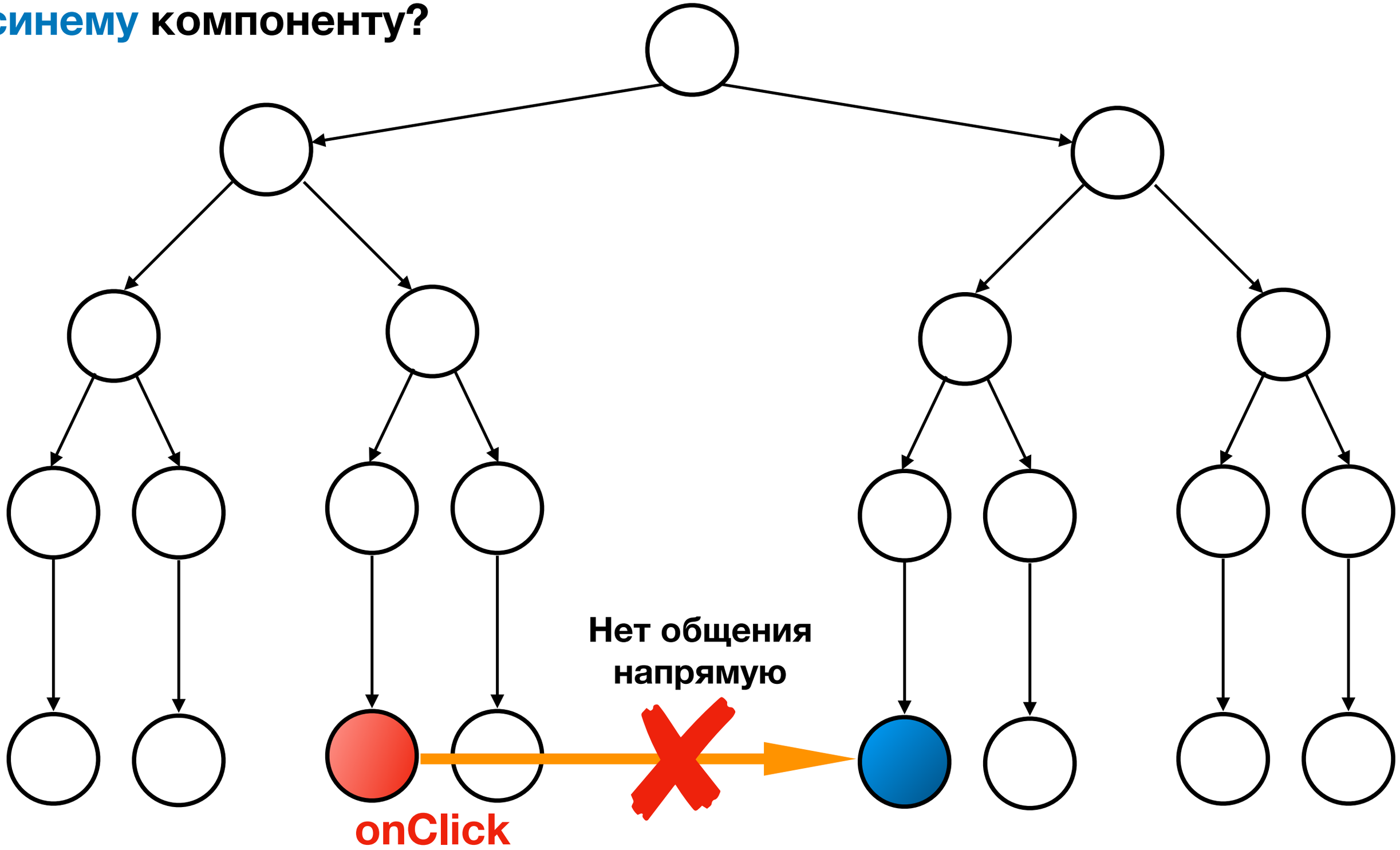
В красном компоненте произошло событие onClick



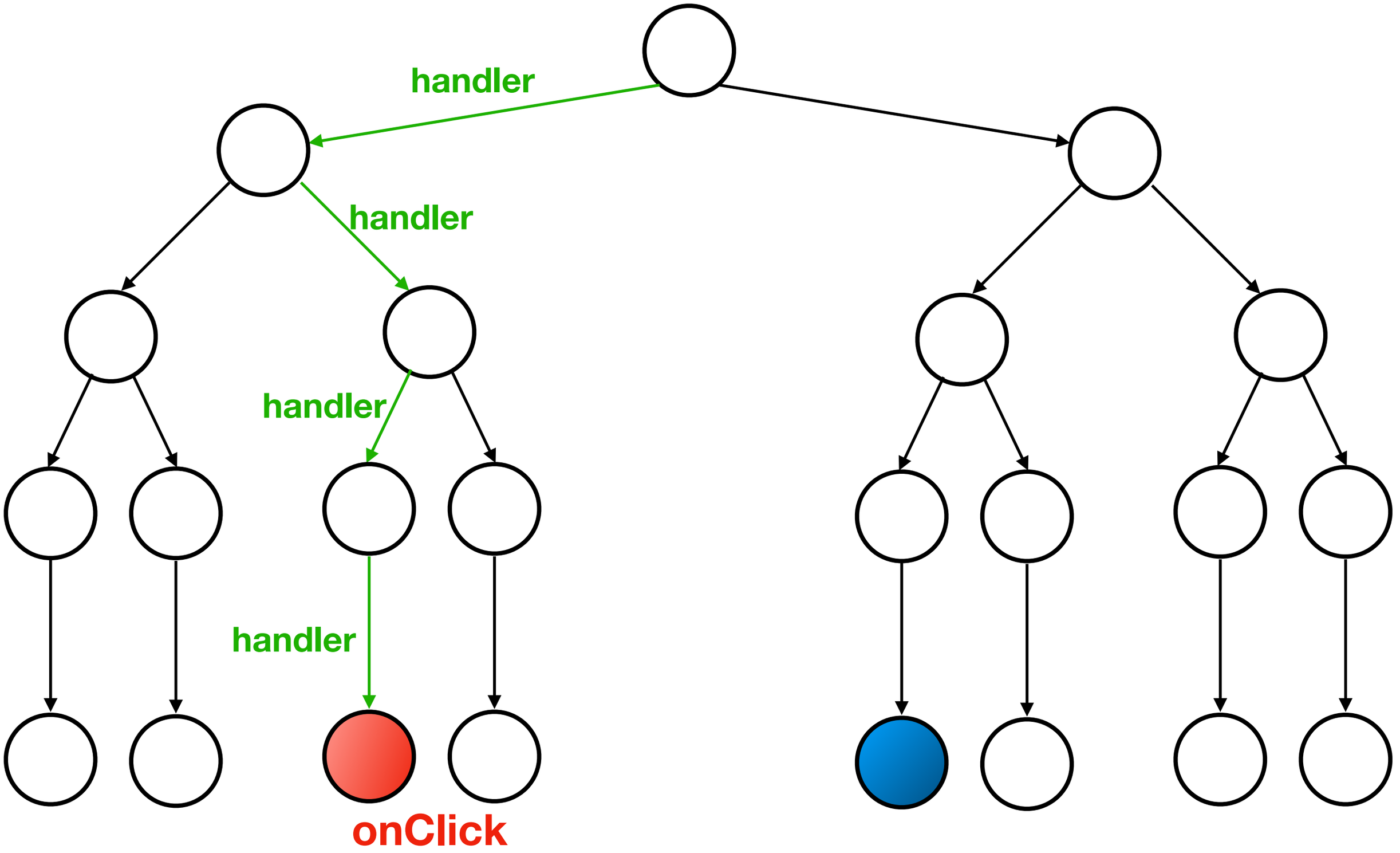
В красном компоненте произошло событие onClick.
Как сообщить об этом
синему компоненту?



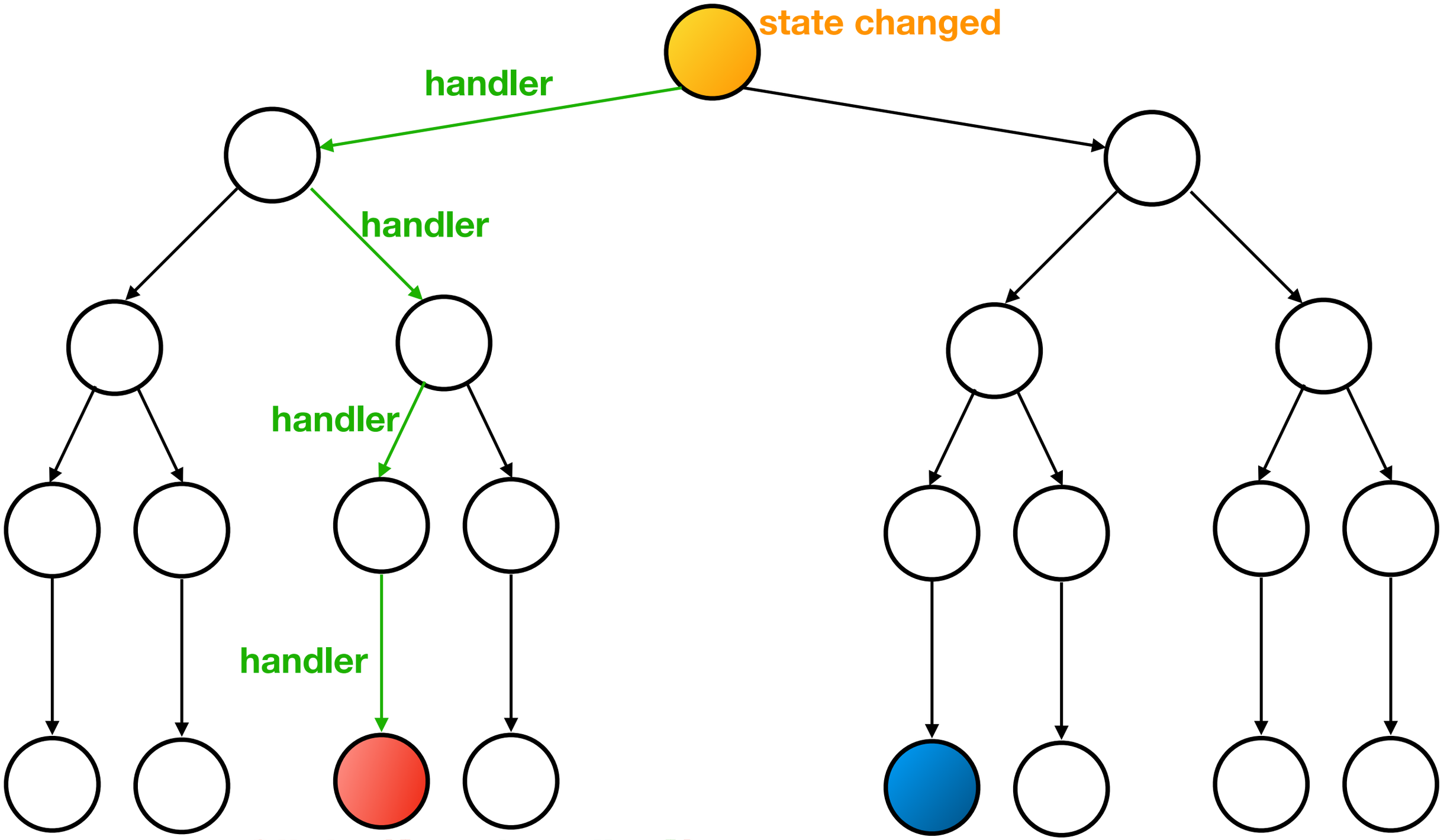
В красном компоненте произошло событие onClick.
Как сообщить об этом
синему компоненту?



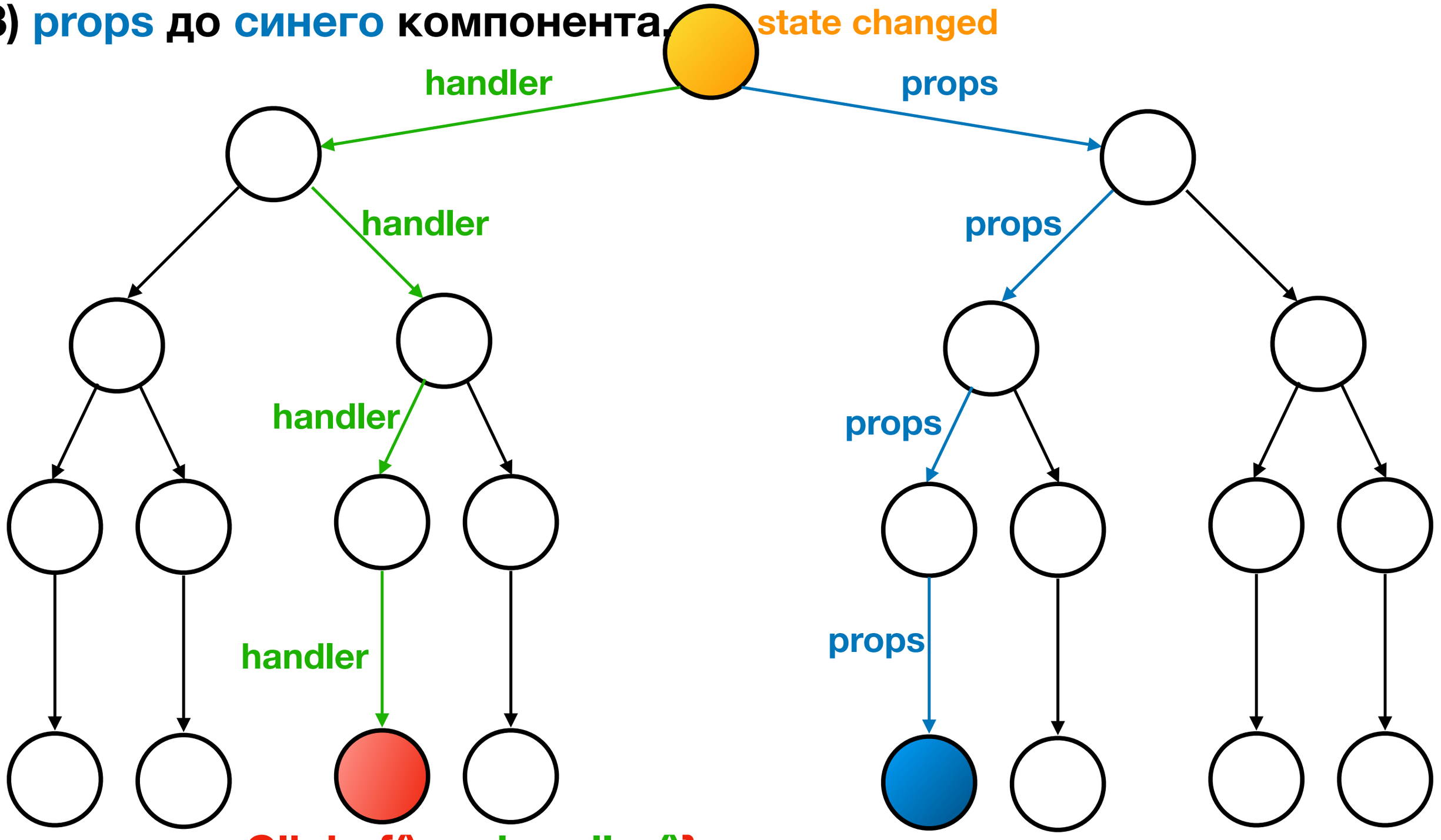
1) Передать функцию handler через props.



1) Передать функцию handler через props.
2) onClick=handler меняет state.

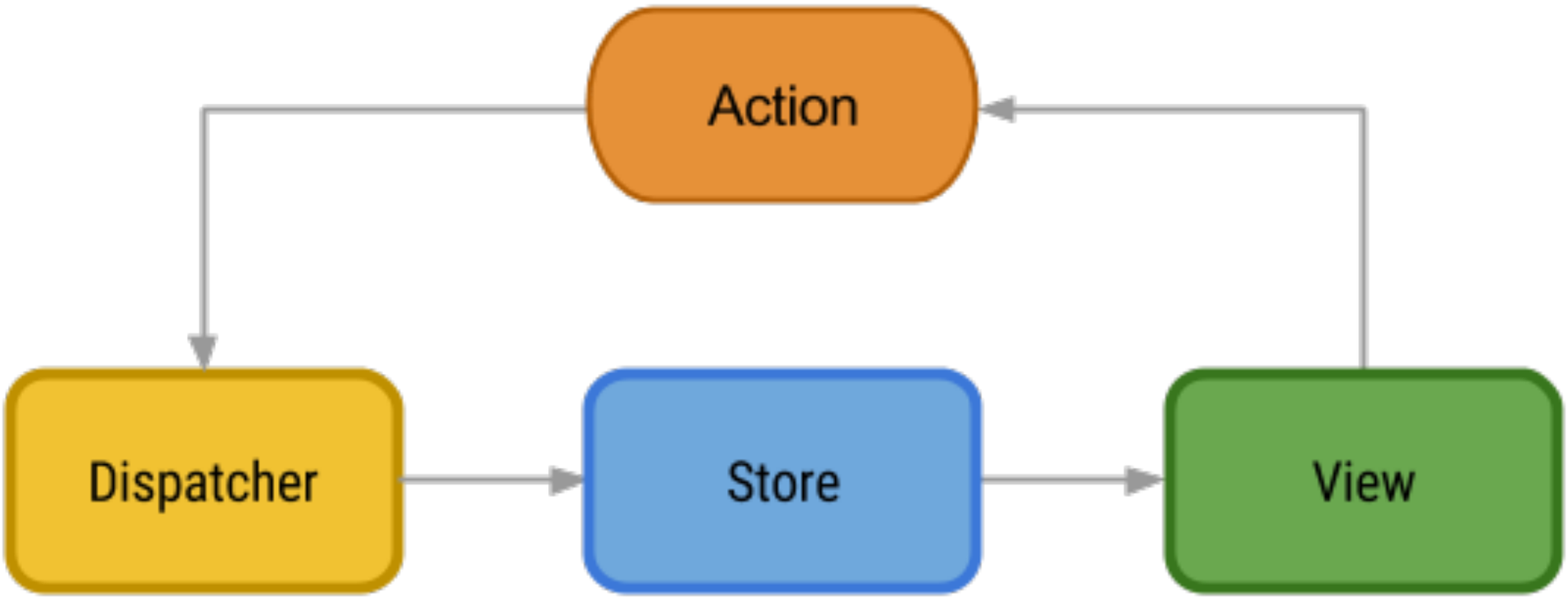


1) Передать функцию handler через props.
2) onClick=handler меняет state.
3) props до синего компонента.

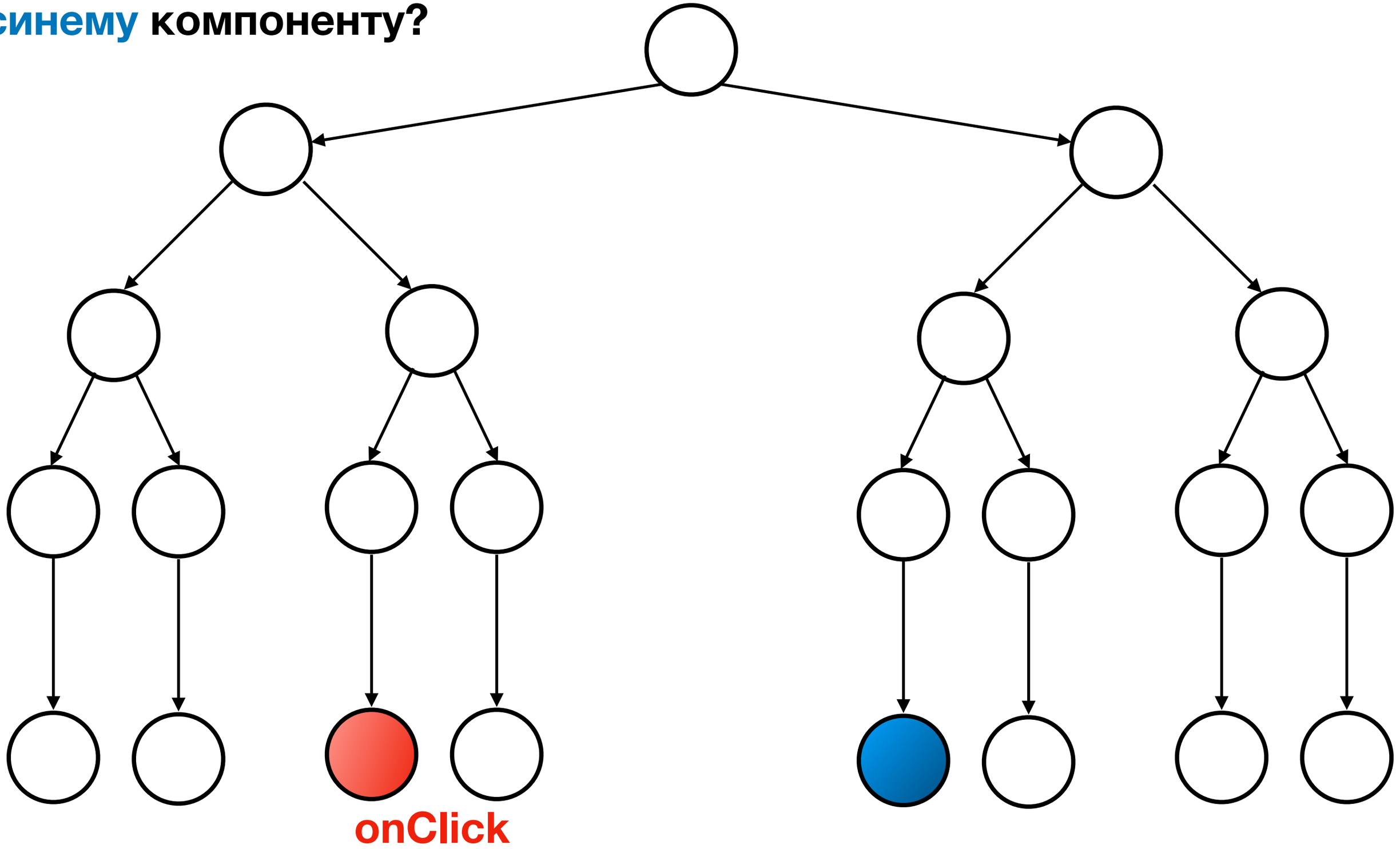


onClick={() => handler()}

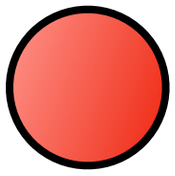
Flux architecture



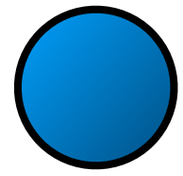
В красном компоненте произошло событие onClick.
Как сообщить об этом
синему компоненту?



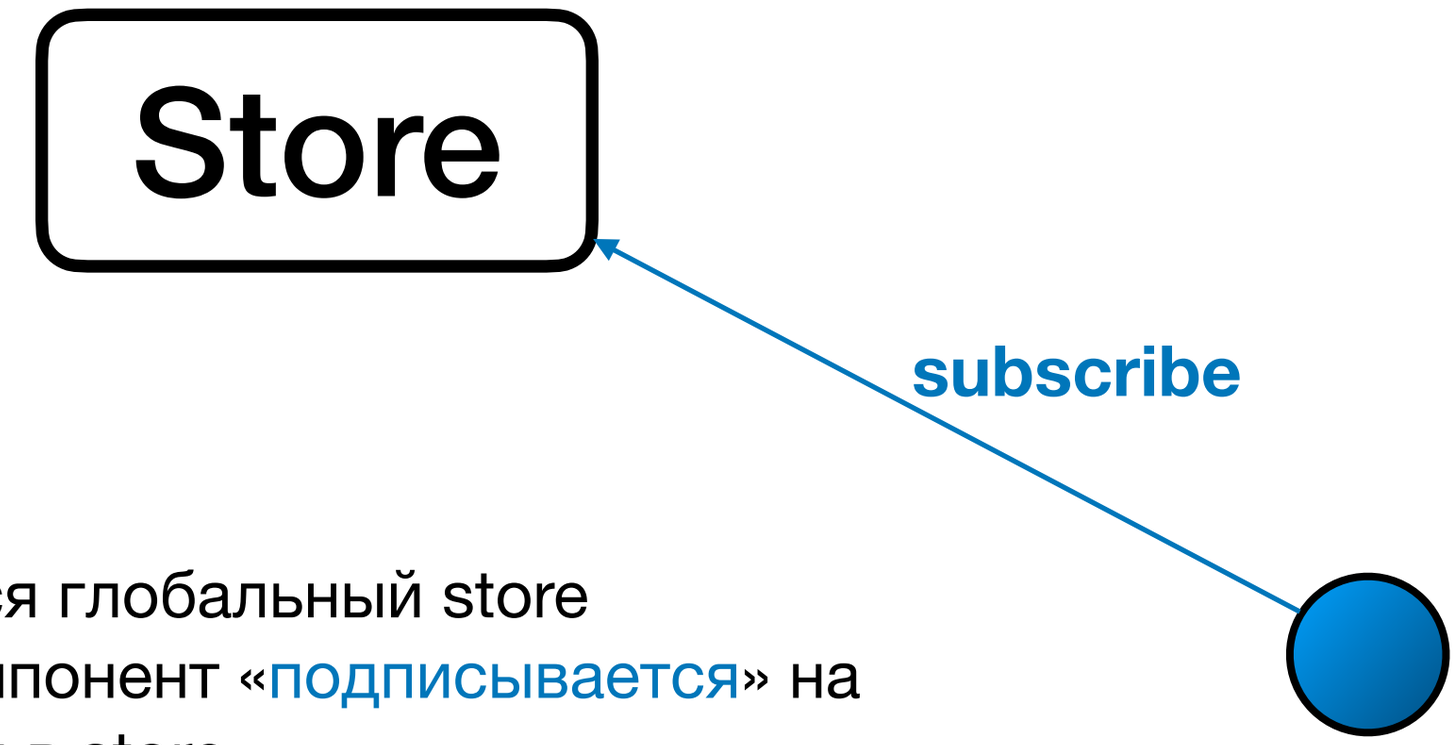
В красном компоненте произошло событие onClick.
Как сообщить об этом
синему компоненту?

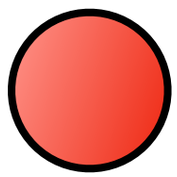


onClick



В красном компоненте произошло событие onClick.
Как сообщить об этом
синему компоненту?





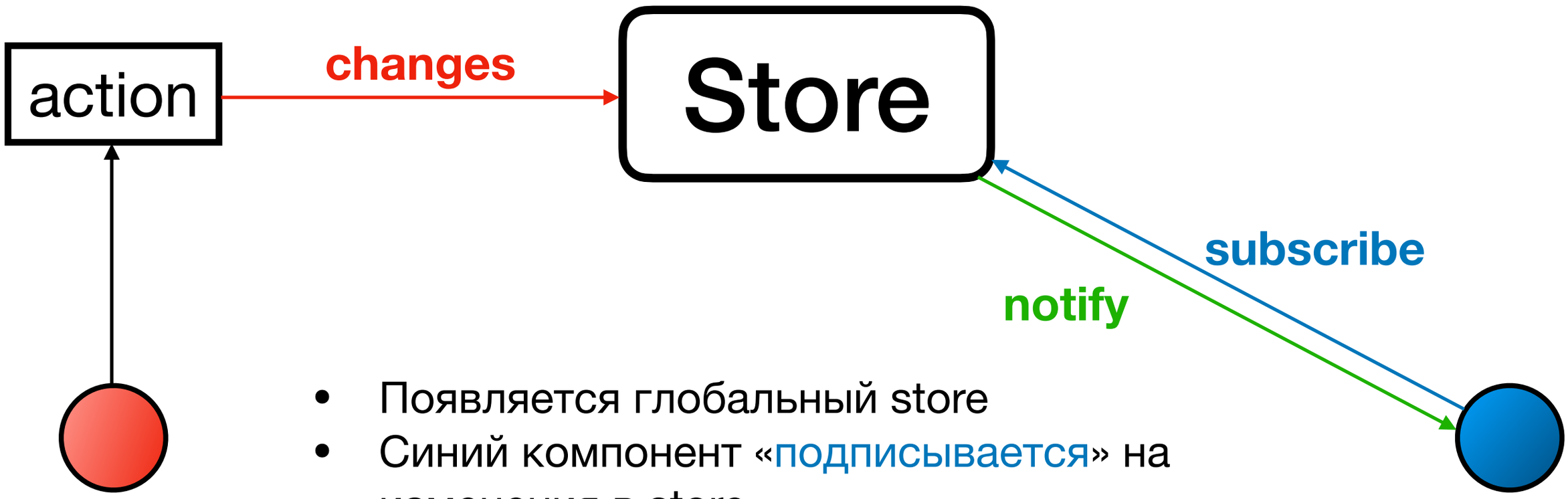
onClick

- Появляется глобальный store
- Синий компонент «подписывается» на изменения в store

В красном компоненте произошло событие onClick.
Как сообщить об этом
синему компоненту?

- Появляется глобальный store
- Синий компонент «подписывается» на изменения в store
- При возникновении onClick красный создает action
- Action вносит изменения в store

В красном компоненте произошло событие onClick.
Как сообщить об этом
синему компоненту?

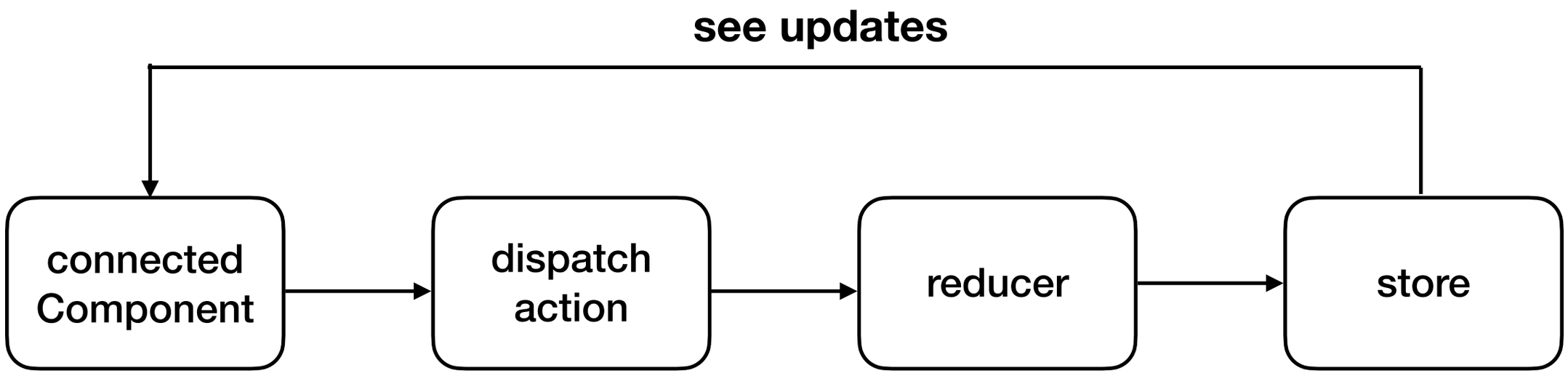


- Появляется глобальный store
- Синий компонент «подписывается» на изменения в store
- При возникновении onClick красный создает action
- Action вносит изменения в store
- Синий сразу же будет «уведомлен» об изменениях в store и перерисуется

Main concepts

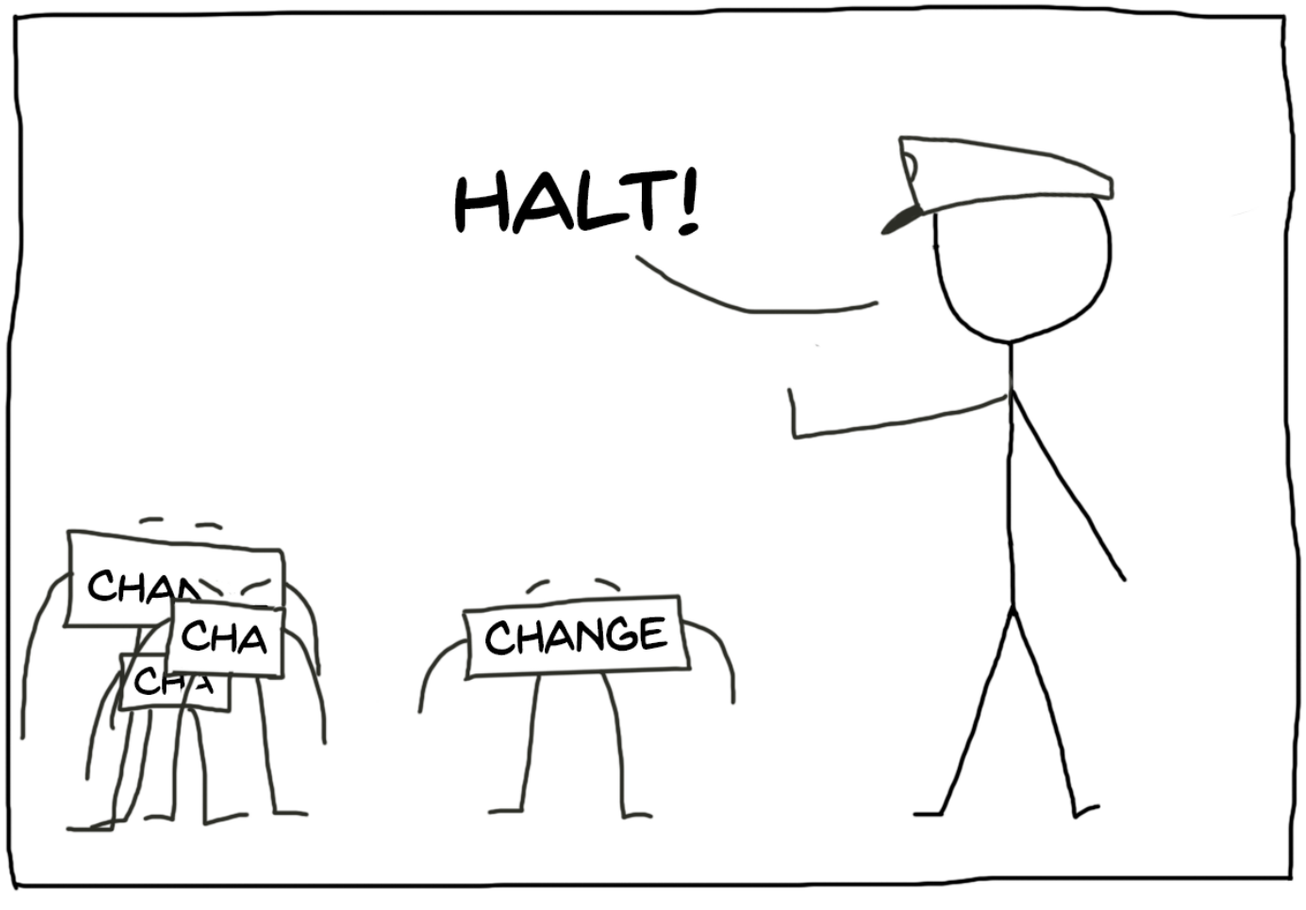
- Store is plain JS object
- Action: plain JS object with a type field that specifies how to change the state
- Reducers: pure functions that take current state and action and return a **new**, immutably updated state
- State is read-only for components, all changes via actions

Main concepts



Immutability pros

- Predictability - references are diff? THERE WAS A CHANGE!
- Change tracking + undo / redo
- Easiness of implementing reactive interface



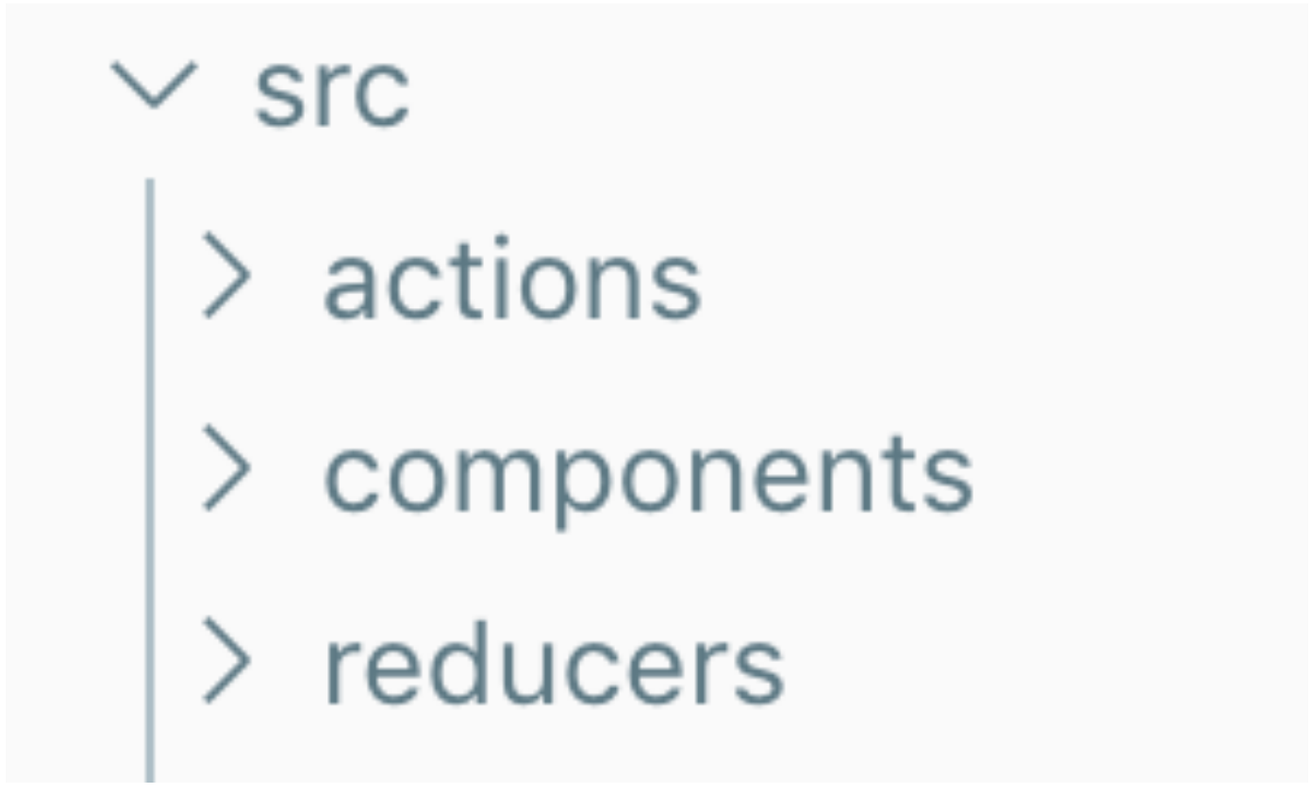
Immutability cons?

```
const car = {
  mark: 'Ford',
  driver: {
    name: 'Valera'
  },
  engine: {
    horsePower: 100
  }
}

const pimpMyRide = car => {
  return {
    ...car,
    engine: {
      horsePower: 200
    }
  }
}

const pimpedCar = pimpMyRide(car);

console.log(car === pimpedCar); // false
console.log(car.engine === pimpedCar.engine); // false
console.log(car.driver === pimpedCar.driver); // true
```

No problems with memory!
Thanks to Shallow Cloning
and Garbage collector

Main concepts

- Store is plain JS object
- Action: plain JS object with a type field that specifies how to change the state
- Reducers: pure functions that take current state and action and return a **new**, immutably updated state
- State is read-only for components, all changes via actions


src
actions
components
reducers

В каком случае класть данные в redux?

- Эти данные могут понадобиться другим компонентам в других частях приложения

- Во всех остальных случаях используете local state