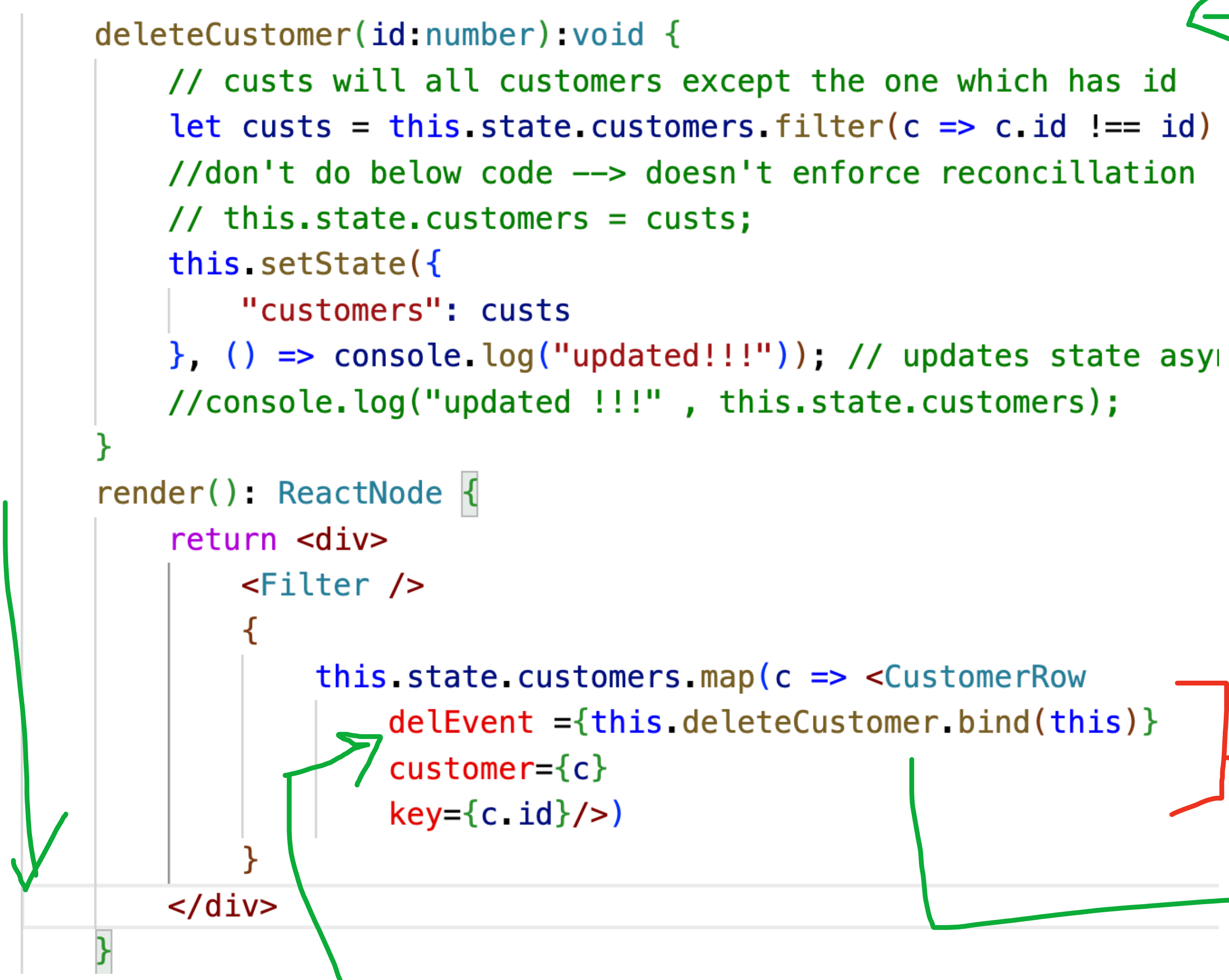
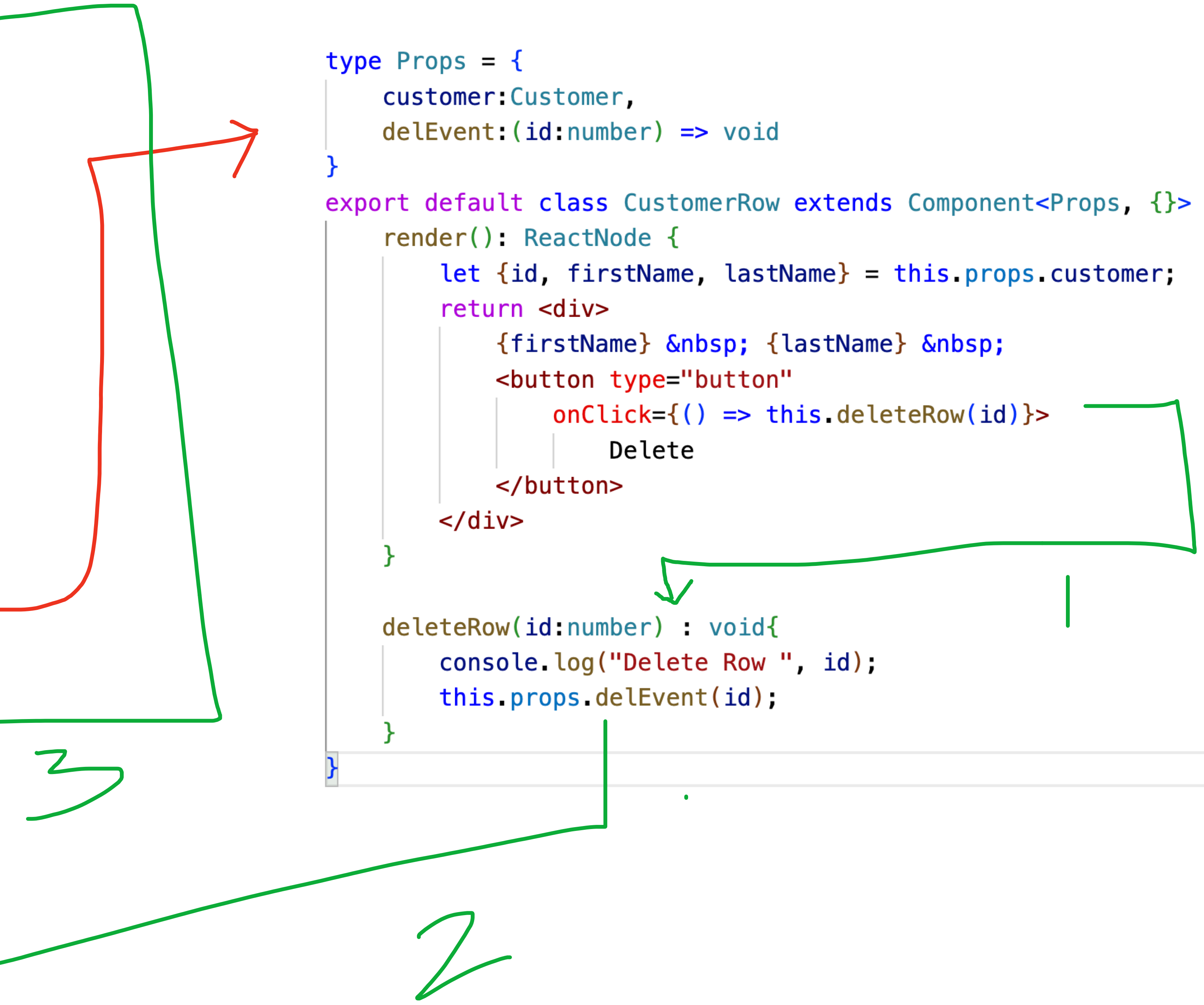
CustomerList



```
deleteCustomer(id:number):void {
    // custs will all customers except the one which has id
    let custs = this.state.customers.filter(c => c.id !== id)
    //don't do below code --> doesn't enforce reconcillation
    // this.state.customers = custs;
    this.setState({
        "customers": custs
    }, () => console.log("updated!!!")); // updates state asy
    //console.log("updated !!!" , this.state.customers);
}
render(): ReactNode {
    return <div>
        <Filter />
        {
            this.state.customers.map(c => <CustomerRow
                delEvent ={this.deleteCustomer.bind(this)}
                customer={c}
                key={c.id}/>)
        }
    </div>
}
```

Whenever state or props chang



```
type Props = {
    customer:Customer,
    delEvent:(id:number) => void
}
export default class CustomerRow extends Component<Props, {}>
    render(): ReactNode {
        let {id, firstName, lastName} = this.props.customer;
        return <div>
            {firstName} &nbsp; {lastName} &nbsp;
            <button type="button"
                onClick={() => this.deleteRow(id)}>
                    Delete
            </button>
        </div>
    }

    deleteRow(id:number) : void{
        console.log("Delete Row ", id);
        this.props.delEvent(id);
    }
}
```

ges --> reconciliation happens

{