State用于储存各个状态，通过在交互中改变State，实现不同状态，同时更新界面，让用户界面和数据保持一致。

当想改变State的值时不能直接使用This.State.key=value

而应该采用seSstate的方法，例如this.setState({key: value});

原因主要有

1. 直接更改state的操作会在你使用setState以后被忽略，使得state的值不符合预期
2. 直接修改state，组件并不会重新触发render(),界面并不会重新渲染

而使用setState 同样有需要注意的地方

setState中存在一个队列来修改State的值，来保证密集调用时的性能，这会使得state的修改不是同步的（要进行到render时才能完成更新），也就是调用时不能确保它是最新的

解决方案目前找到两种：

1. 使用回调函数来获取最新的State，例如

this.setState((prevState, props) => ({

counter: prevState.counter + props.increment

}));

回调函数会在setState完成后调用保证state的值是最新的

1. 在setTimeout中使用setState也能保证state的值是最新的，原理不清楚
2. ~~此外还有一种promise的方法，看不懂~~

以下是*每秒更新state来实现时间同步的代码*

class App extends Component { {*/\* 每秒更新state来实现时间同步 \*/*}

constructor(props) {

super(props);

this.state = {date: new Date()};

}

componentDidMount() {

this.timerID = setInterval(

() => this.tick(),

1000

);

}

componentWillUnmount() {

clearInterval(this.timerID);

}

tick() {

this.setState({

date: new Date()

});

}

render() {

return (

<div>

<h2>北京时间 {this.state.date.toLocaleTimeString()}.</h2>

</div>

);

}

}

不懂的地方：

对使用Props来传递参数的原理不清楚，父级子级之间调来调去的调发也不是很清楚