setState 異步執行

狀態在進行改變時，React會進行合併與最佳化等內部工作，必會有異步執行的特性

```
1 <h1
2   onClick={() => {
3     setTotal(total + 1) // 呼叫要變動狀態的方法
4     console.log(total) // 想要得到變動後的狀態
5   }}
6 >
7   {total}
8 </h1>
```

絕對得不到變動後的狀態，因第3行的setState(或setXXX)方法是異步執行(比第4行還晚執行)

setState 異步執行 - 對應策略

策略一：先用變數接住最後更動的值 (推薦！)

先運算出最後會變動的狀態值是什麼，
接著再用它作其它的處理，例如設定給 React 狀態，或是傳到伺服器、輸出在主控台...等

```
1 <h1
2   onClick={() => {
3     const newTotal = total + 1 // 先運算出變動後結果值
4     setTotal(newTotal) // 此處呼叫要變動狀態的方法
5     console.log(newTotal) // 此處想要得到變動後的狀態
6   }}
7 >
8   {total}
9 </h1>
```

setState 異步執行 - 對應策略

策略二：componentDidUpdate生命周期方法

```
class App extends React.Component {
  constructor() {
    super()
    this.state = { total: 0 }
  }

  componentDidUpdate() {
    // 在total有改變後，取得total
    console.log(this.state.total)
  }

  render() {
    return (
      <h1
        onClick={() => {
          // 此處呼叫要變動狀態的方法
          this.setState({ total: this.state.total + 1 })
        }}
      >
        {this.state.total}
      </h1>
    )
  }
}
```

setState 異步執行 - 對應策略

策略二：componentDidUpdate生命周期方法

```
1  function App() {
2    const [total, setTotal] = useState(0)
3
4    // 模擬生命周期方法 componentDidUpdate
5    useEffect(() => {
6      console.log(total) // 在total有改變後，取得total
7    }, [total])
8
9    return (
10     <h1
11       onClick={() => {
12         setTotal(total + 1) // 此處呼叫要變動狀態的方法
13       }}
14     >
15       {total}
16     </h1>
17   )
18 }
```

setState 異步執行 - 對應策略

策略三：使用setState第二個傳入參數值(為callback函式)，只有類別型元件可以這樣用

```
<h1
  onClick={() => {
    // 此處呼叫要變動狀態的方法
    this.setState(
      { total: this.state.total + 1 },
      () => {
        // 在total有改變後，取得total
        console.log(this.state.total)
      }
    )
  }}
>
  {this.state.total}
</h1>
```

setState方法第二個傳入參數值，在更動狀態完成後會進行呼叫 (只有類別型元件可以使用)