create-react-app project-name

* 創立專案

npm start

* 在專案底下輸入，可以啟用專案

JSX語法中，想要用自己的組件開頭一定要大寫

**## DOM**

**## 1. 有State數據**

**## 2. 有JSX模板**

**## 3. 數據+模板結合，生程實體的DOM，來顯示**

**## 4. State發生改變**

**## 5. 數據+模板結合，生程實體的DOM，替換原始的DOM**

**## 缺點:**

**##  每次都要生成一個完整的DOM，非常耗性能，且替換原始的DOM也耗性能**

**## DOM改良**

**## 1. 有State數據**

**## 2. 有JSX模板**

**## 3. 數據+模板結合，生程實體的DOM，來顯示**

**## 4. State發生改變**

**## 5. 數據+模板結合，生程實體的DOM，不直接替換原始DOM**

**## 6. 新DOM和原始DOM做比對，找差異**

**## 7. 找出變化地方**

**## 8. 只使用新DOM中的變化元素，去替換原始DOM的元素**

**## 比較:**

**## 雖然改良步驟較多，但整個DOM替換所需性能較大，比對後替換會相對好一點，但不多**

**## React 虛擬DOM方案**

**## 1. 有State數據**

**## 2. 有JSX模板**

**## 3. 數據+模板結合，生成虛擬DOM(一個JS的描述，一個Object或陣列，用來描述實體的DOM)**

**##    ['div' , {id : 'abc'} , ['sapn' , {} , 'hello world']]**

**## 4. 使用虛擬DOM，生程實體的DOM，來顯示**

**##    <div id='abc'><span>hello world</span></div>**

**## 5. State發生改變**

**## 6. 生成新的虛擬DOM**

**##    ['div' , {id : 'abc'} , ['sapn' , {} , 'bye bye']]**

**## 7. 比較虛擬DOM的差異**

**## 8. 直接改便原始DOM的差異地方**

**## 比較:**

**## 在JS中生成一個JS描述，所需性能是極小的，所以在第6步用虛擬DOM取代了實體DOM的生成，提升極大性能**

**## 且在第7步JS之間的比較也比DOM之間的比較性能損耗大大降低**

生命週期函數:指在某一個時刻會自動執行的函數

**component創建**

componentWillMount : 當組件**第一次**即將被渲染至頁面時執行。

componentDidMount : 當組件**第一次**被渲染至頁面後執行。

**State更改**

shouldComponentUpdate : 當組件要被更新前執行，用來確認此次資料是否要更  
 新return false系統將不更新。(須返回Boolean)。

componentWillUpdate : 當組件更新前被執行，但他在shouldComponentUpdate  
 之後，且shouldComponentUpdate要返回true。

componentDidUpdate : 當組件更新後執行。

**Props更改**

componentWillReceiveProps : 當子組件從父組件接收參數後，父組件的render  
 被重新執行後執行。

shouldComponentUpdate : 同上。

componentWillUpdate : 同上。

componentDidUpdate : 同上。

**component刪除**

componentWillUnmount : 當組件即將被移除時執行。