

React 入門實做家教班

KAI Coding Lab

WIFI :  65892091

投影片 :

大綱

熟悉library語法架構

Part1

react.js 原理

react.js 使用優點

React 最基本使用範例 (JSBIN練習)

NPM介紹

終端機常用指令介紹

VSCode常用功能介紹

利用create-react-app 快速建立 react 專案

講解資料夾內檔案

Node 基礎知識講解

JSX 如何使用

專案時間：第一個實作

必須熟悉語法

state介紹

專案時間：利用React 做一個Todo List

優化&深入

Part2

ant-design教學

專案時間: 將Todo-List套用ant-design

State v.s. Props

父傳子vs子傳父

小練習時間：練習元件props

專案時間：將todo list做folder structure

redux學習-為何要用redux

redux學習-熟悉語法

實作：todo list + redux

結合資料庫+大量實作

Part3

Firebase 教學

專案時間：firebase+todo-list

專案實作：聊天室

專案實作：聊天室進階版

npm run build

React -

facebook 官方所維護的開放原始碼 JavaScript 函式庫
可以降低互動式網頁應用程式開發難度
自動處理各種複雜 UI 組件與資料間的連動關係
改善應用程式執行效能。

react.js 三大特點

1. 設計元件，把邏輯端跟html寫在一起

Javascript

```
/**  
 * index.html  
 */  
<div id="output"></div>  
  
/**  
 * main.js  
 */  
document.getElementById('output').innerHTML="hello, world"
```

React.js

```
/*  
 * HelloWorld.js  
 */  
function App() {  
  const [value, setValue] =  
    React.useState('hello, world')  
  render() {  
    return <div>{value}</div>;  
  }  
}
```

react.js 三大特點

2.一律重繪

一個application的狀態會非常多，而且會頻繁的變化。

如果我們監聽某一個事件，就要針對這一個狀態進行 DOM 的處理，越來越複雜。

React 中：

1. 改狀態的資料 (state更新)
2. 自動進行更新畫面

簡單多了，是吧！

但每次重繪，不會很耗效能嗎？

功能越多，觸發頁面的事件越多，前端資料狀態的變化也越頻繁，因此改變 DOM 結構也會越頻繁。我想你應該要知道的是 HTML DOM 是一個很大的樹狀資料，每當一個事件發生，而需要進行新增、修改、刪除時，底層的程式就必須先從這一棵大樹中，一直深入，再深入，直到找到某一片你有興趣的葉子，可想而知這是多麼大的工程！

所以，為了要解決

1. 頻繁改變 DOM 是很耗效能的
2. 前端程式很難封裝和模組化

react.js 三大特點

3. Virtual Dom

並不會每次都重繪節點的內容
因為react會去判斷舊狀態跟新狀態
如果有更動，才會去改變真實節點的內容。

開發者跟 DOM 中間有一個功能
能幫我們改變最小幅度的 DOM。
React 會在根據舊狀態和新狀態進行比對，
將確切有需要更動的 DOM 真實反映在瀏覽器的 DOM 上。

react.js 使用優點

1. 好維護，輕架構
2. 很多開發者，很多職缺
3. 能進階寫react native (android&ios)
4. 2013最初版本出來，React 16新版hooks在2018年末出來

React 最基本使用範例

A screenshot of the jsbin.com interface. The top navigation bar shows a search bar with '不安全 | jsbin.com/?html,js,output' and various bookmarks. The main area has tabs for 'HTML', 'CSS', 'ES6 / Babel', 'Console', and 'Output'. The 'HTML' tab is selected, displaying the following code:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>JS Bin</title>
</head>
<body>
<script src="https://fb.me/react-15.1.0.js"></script>
<script src="https://fb.me/react-dom-15.1.0.js"></script>

<div id='example'></div>

</body>
</html>
```

The 'ES6 / Babel' tab contains the following code:

```
ReactDOM.render(
  <h1> Hello, world! </h1>,
  document.getElementById('example')
);

// document.getElementById('example').innerHTML='<h1> Hello, world! </h1>'
```

The 'Output' panel shows the result: **Hello, world!**.

```
<script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
```

A screenshot of the jsbin.com interface. The top navigation bar shows a search bar with 'Hello, Kai' and various bookmarks. The main area has tabs for 'HTML', 'CSS', 'JSX (React)', 'Console', and 'Output'. The 'JSX (React)' tab is selected, displaying the following code:

```
function App() {
  const [name, setName] = React.useState('Kai');

  return(
    <div>Hello, {name}</div>
  )
}

ReactDOM.render(
  <App/>,
  document.getElementById('example')
)
```

The 'Output' panel shows the result: **Hello, Kai**.

NPM介紹

javascript套件庫

安裝新版Node.js會自動安裝 (<https://nodejs.org/zh-cn/download/>)

常用指令：

看版本號：**node -v && npm -v**

在專案內安裝：**npm install XXX** (套件名稱)

全域安裝：**npm install -g XXX** (套件名稱)

執行package.json中的scripts：**npm run YYY** (scripts名稱)

上面指令皆在終端機中輸入。

終端機常用指令介紹

進入某個資料夾：**cd XX(資料夾名稱)**

回到上層資料夾：**cd ..**

查看這個資料夾內有什麼檔案：**ls**

我現在在哪層資料夾內：**pwd**

建立資料夾：**mkdir XX(資料夾名稱)**

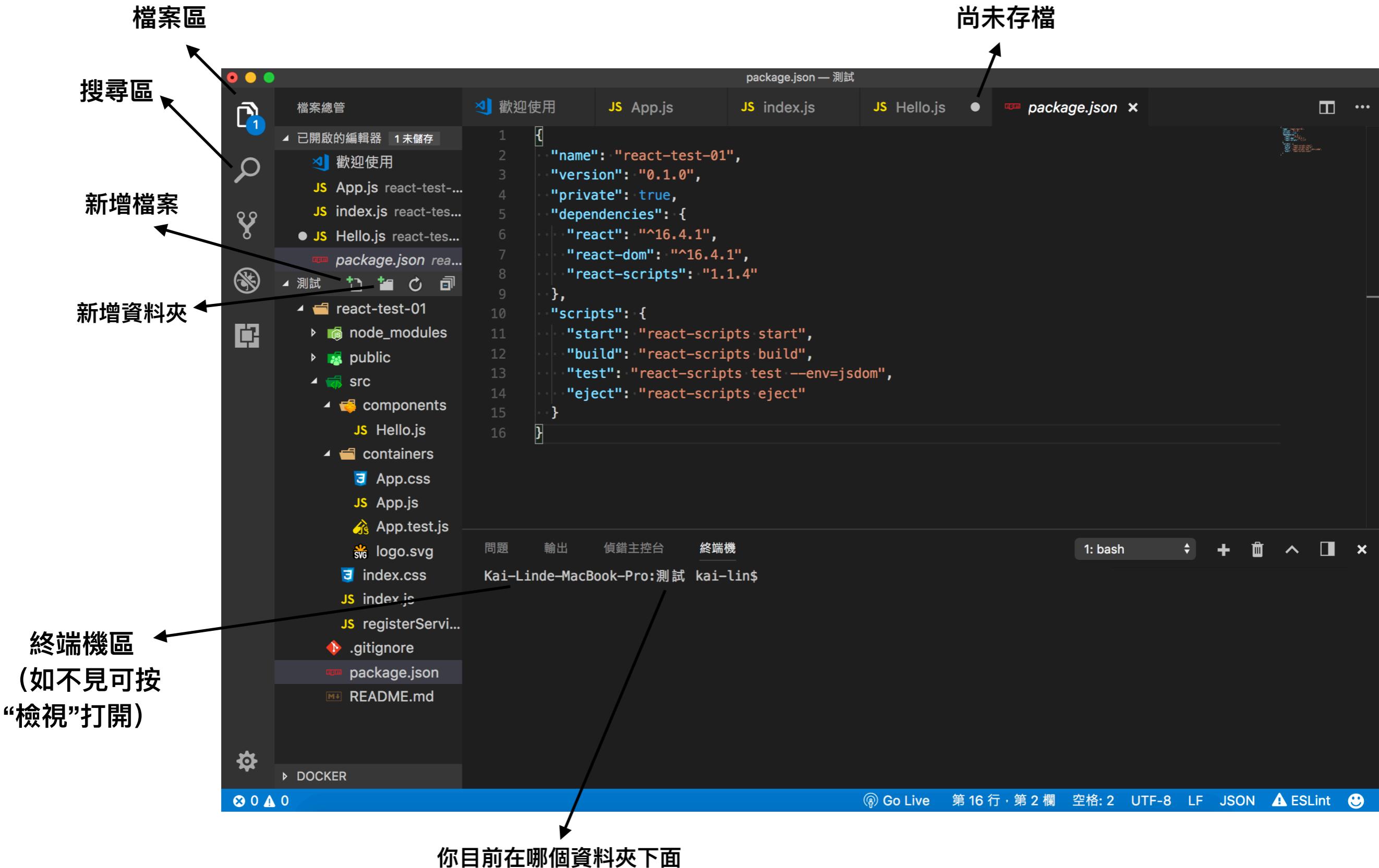
刪除全部：**rm -rf XX(名稱)**

建議編輯器：**VSCode**（編輯器中有內附終端機）

安裝：<https://jeasonstudio.gitbooks.io/vscode-cn-doc/content/md/%E7%BC%96%E8%BE%91%E5%99%A8/%E5%AE%89%E8%A3%85.html>

先在桌面成立一個資料夾「react.js入門實作」
並在vscode中開啟

VSCode常用功能介紹



利用create-react-app 快速建立 react 專案

1. 新增一個資料夾react練習區，移至vscode中開啟
2. `npm install -g create-react-app`
3. `create-react-app first-practice` (後面是專案名稱)
4. `cd first-practice` (專案名稱打到一半，按tab，就會將名稱全部打出來)
5. `npm start` (執行程式碼)

講解資料夾內檔案

node_modules => 套件放置區

package.json=> dependencies=>裝了哪些套件

=> scripts =>指令快捷鍵

=> name =>名稱

=> version =>版本

npm run eject

注意：這是單向操作。有一次eject，你不能回去！

如果您對構建工具和配置選擇不滿意，您可以eject隨時進行。此命令將從項目中刪除單個構建依賴項。

相反，它會將所有配置文件和傳遞依賴項（Webpack，Babel，ESLint等）複製到項目中，以便您可以完全控制它們。除了eject仍然有效之外的所有命令，但它們將指向複製的腳本，以便您可以調整它們。在這一點上，你是獨立的。

Public => 放置公開檔案（圖片/html版...）

Src => source code（程式碼）

Node 基礎知識

Import XX from YY

將YY名稱的套件引入，並設一個變數XX在此檔案中使用

```
import React from 'react';
import ReactDOM from 'react-dom';
```

將YY檔案引入，並設一個變數XX在此檔案中使用

```
import App from './App';
import registerServiceWorker from './registerServiceWorker';
```

#備註：./代表同層檔案，../代表上層檔案， ../../代表上兩層檔案

Import {ZZ} from YY

將YY名稱的套件裡的ZZ引入，並設一個變數ZZ在此檔案中使用

```
import React, { Component } from 'react';
```

Import YY

將YY檔案引入，例如css檔

```
import './index.css';
```

export default

將此檔案名稱為App的函數或class匯出，讓別個檔案使用。

```
export default App;
```

在開始寫程式碼之前...

React 慣例（舊版 React v15之前）

```
class App extends Component {  
    state={  
        name:"",  
        input:""  
    }  
    componentDidMount=()=>{  
    }  
    hello=()=>{  
        this.setState({name:this.state.input})  
    }  
    handleChange=(e)=> {  
        this.setState({ input: e.target.value });  
    }  
    render() {  
        return (  
            <div className="App">  
                <input onChange={this.handleChange}></input>  
                <button onClick={this.hello}>click</button>  
                <div>  
                    即時顯示 : {this.state.input}  
                </div>  
                <div>  
                    你輸入的是 : {this.state.name}  
                </div>  
            </div>  
        );  
    }  
}
```

初始化state

function執行

渲染的JSX（似html）

React 慣例 (新版 React v16)

```
function App () {  
  const [name, setName] = useState('')  
  const [input, setInput] = useState('')  
  
  useEffect(()=>{  
    //當有改變下面的state，會跳出的  
    return () => {  
      };  
    }, [])  
  
  const hello=>{  
    setName(input)  
  }  
  
  return (  
    <div className="App">  
      <input onChange={e => setInput(e.target.value)} />  
      <button onClick={hello}>click</button>  
      <div>  
        即時顯示：{input}  
      </div>  
      <div>  
        你輸入的是：{name}  
      </div>  
    </div>  
  );  
}
```

初始化state

function執行

渲染的JSX (似html)

開始寫程式碼GO

(first-practice)

先讓我們清空<div>內的程式碼
引入react中的**useState,useEffect**

```
import React,{useState,useEffect} from 'react';
```

在function中加個**const [XXX, setXXX] = useState('')**

並在JSX中使用**{XXX}**來呈現

初始值
(可以是字串/數字/array/object)

接下來讓我們練習在function App中加個**function**

```
const changeIt=()=>{  
}
```

並加上一個**button**，去執行。

(console.log顯示有執行)

```
<button onClick={changeIt}>click</button>
```

接下來讓我們在function中去改變**state**的值。

利用setXXX('i change it.')

state是一個狀態，把會更新的資料放到**state**
所有資料流=>更改**state**=>自動更新畫面（還會自動執行useEffect）

不會重新更新

來試試初始值設為數字的計數器吧！

牛刀小試

1. 請增加減數器按鈕
2. 請增加將文字消失的按鈕（將文字變為空值）

恭喜你完成
第一次操作react的state資料流....

現在讓我們來了解JSX和React必須熟悉用法吧！

JSX 簡介

JSX 是一種 JavaScript 的擴充語言，加入了一些 HTML 標籤的語法

```
const element = <h1>Hello, world!</h1>;
```

→ 拿來render到html

React 架構在設計上就將 HTML 標籤與 JavaScript 控制邏輯合併

以 JSX 來描述 UI 的外觀與運作邏輯

打造出 React 的 UI 組件 (components)

再用這些 UI 組件堆疊出個應用程式。

JSX 如何使用

1. HTML 的 class 屬性在 JSX 須寫為 className (class 為 JSX 保留字)
2. 夾在元件標記或HTML的DOM元素標記的JavaScript程式碼時，要使用大括號{}框住

```
<div className="App">
  <header className="App-header">
    <img src={logo} className="App-logo" alt="logo" />
    <h1 className="App-title">Welcome to React</h1>
  </header>
  <p className="App-intro">
    To get started, edit <code>src/App.js</code> and save to reload.
  </p>
</div>
```

```
<ul>
  {
    this.state.items.map((value, index) => {
      return <TodoItem key={index} text={value} index={index}>
        <button onClick={this.handleRemoveItem}></button>
      </TodoItem>
    })
  }
</ul>
```

3. 同 JS，註解可以用 /* */ 或 //，在 tag 中使用的話則須用大括號 {} 包住
4. 事件觸發是採用駝峰式命名法而不是全部小寫。

EX :

```
<button onClick={myFunction}></button>
```

5. style 屬性要以 JS 物件的格式設定 (JSON)，採用駝峰式命名法而非-，數值的單位是 px，其他單位要用單引號包住 (EX: '50%')。別忘記外面要再加上一層大括號。

EX :

```
<a style={{ fontSize: '16px', color: '#FF0' }}>hello</a>
```

JSX 如何使用

6. JSX語法中只能有一個根元素

```
// 錯誤示範!!
ReactDOM.render(
  <div>Test</div>
  <div>Test 2</div>,
  document.getElementById('root')
)
```

```
//正確範例
ReactDOM.render(
  <div>
    <div>Test</div>
    <div>Test 2</div>
  </div>,
  document.getElementById('root')
)
```

JSX 如何使用

7. JSX常用用法整理

```
<div>你輸入的是 : {name}</div>
```

在html tag內容加上變數

```
<input value={input} onChange={e => setInput(e.target.value)} />
```

```
<button onClick={changeIt}>Change</button>
```

tag的屬性塞入變數 (state/props/function)

```
<input value={initialValue}/>
```

```
{props.myBag.map((thing)=>(  
  <div>{thing}</div>  
>))}
```

迴圈 (使用map=>相當於forEach)

```
<Person name={window.isLoggedIn ? window.name : ''} />
```

三元運算 (相當於if else)

```
/* child comment, put {} around */
```

放備註

...

React 必須熟悉用法 (背)

state使用

初始化

```
const [title, setTitle] = useState('')  
const [myBag, setMyBag] = useState([])  
const [number, setNumber] = useState(0)
```

存入

```
setTitle('')  
setMyBag(myBag.push('new thing'))  
setNumber(number+1)
```

讀取 (在JSX中)

```
{title}, {number}
```

function使用

```
function hello(){  
}  
  
<button onClick={handleClick}>click</button>
```

不帶變數：

```
const handleClick = () => {  
  setName(input)  
}
```

帶變數：

```
const handleClick = (para) => {  
  setName(para)  
}
```

```
<button onClick={()=>handleClick('hello')}>click</button>
```

如果是內建的變數，就不需要

useEffect使用(call api時放的地方=>初始化)

css使用

```
<div style={{backgroundColor:'red'}}>  
  </div>
```

```
useEffect(()=>{  
  //當有改變下面的state，會執行的  
  return () => {  
    };  
}, [state])
```

所有資料流=>更改state這個物件=>自動更新畫面

專案時間：第一個實作

```
function App() {  
  const [name, setName] = useState("")  
  const [input, setInput] = useState("")  
  
  const handleClick = () => {  
    setName(input)  
  }  
  
  const handleChange = (event) => {  
    setInput(event.target.value)  
  }  
  
  return (  
    <div className="App">  
      <input onChange={handleChange}></input>  
      <button onClick={handleClick}>click</button>  
      <div>  
        即時顯示 : {input}  
      </div>  
      <div>  
        你輸入的是 : {name}  
      </div>  
    </div>  
  );  
}
```

HTML input onchange
獲得變化值的方法

hello click
即時顯示 : hello
你輸入的是 :

hello click
即時顯示 : hello
你輸入的是 : hello

牛刀小試

小明

即時顯示：小明

click

你輸入的名字：小明

click

好心情對話：小明好棒喔！

牛刀小試2

小明 小美

即時顯示：小明

click

你輸入的名字：小明

click

好心情對話：小明好棒喔！

click

羞羞臉對話：小明愛小美

(todo-list)

專案時間：利用React 做一個Todo List

- 1.動態再換
- 2.SPR

React Todo List

- 倒垃圾
- 回覆email
- 種一棵樹
- 跑步3000m
- 計劃新目標

2未完成/5總數

新增任務

恭喜你完成第一個專案

來利用ant-design(UI framework)讓你的網站更漂亮吧！

ant-design套用方法

終端機中：

npm install antd

修改 src/App.css，在文件頂部引入 antd/dist/antd.css。

@import '~antd/dist/antd.css';

在程式碼中：

import { Button,Input,Checkbox } from 'antd';

把<button></button>改成<Button></Button>

(todo-list)

專案時間: 將Todo-List套用ant-design



人生最幸福的兩件事....

- 1.自己暗戀的人也在偷偷暗戀著自己
- 2.寫完一個專案竟然沒出現bug

元件間溝通：State v.s. Props

Props

- are immutable
 - which lets React do fast reference checks
- are used to pass data down from your view-controller
 - your top level component
- have better performance
 - use this to pass data to child components

State

- should be managed in your view-controller
 - your top level component
- is mutable
- has worse performance
- should not be accessed from child components
 - pass it down with props instead

State=>同一元件的狀態管理

Props=>父子元件的狀態管理

元件間溝通 : State v.s. Props

app.js

```
return (  
  <div>  
    {name}  
  </div>  
)
```

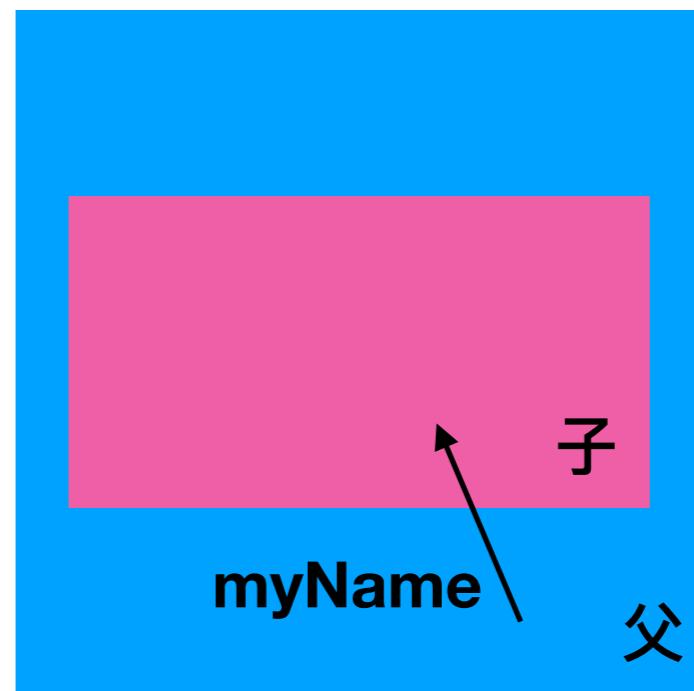


Import MyName from './MyName'

```
return (  
  <div>  
    <MyName name={name}/>  
  </div>  
)
```

MyName.js

```
function MyName(props){  
  return (  
    <div>名字DIV : {props.name}</div>  
  )  
}  
....  
export default MyName
```



元件間溝通：State v.s. Props

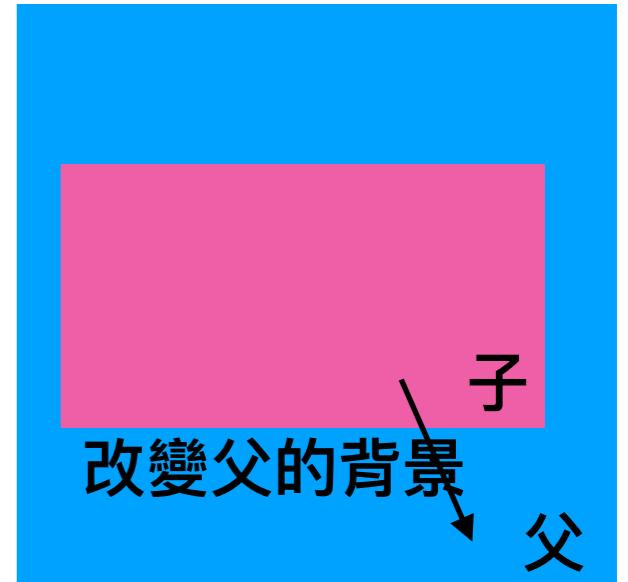
父傳子 如上頁

子傳父 **ChangeBGColorBtn.js**

```
function ChangeBGColorBtn(props){  
  return (  
    <div>  
      <button onClick={()=>props.changeFn('blue')}>click</button>  
    </div>  
  )  
}
```

app.js

```
Import MyName from './MyName'  
....  
  
const [color,setColor] = useState('white')  
  
const changeFn=(color)=>{ setColor(color) }  
  
return (  
  <div style={{backgroundColor:color}} >  
    <ChangeBGColorBtn changeFn={changeFn} />  
  </div>  
>;
```



(props-practice)

小練習時間：

成立一個小專案props-practice來練習元件props
父傳子，子傳父

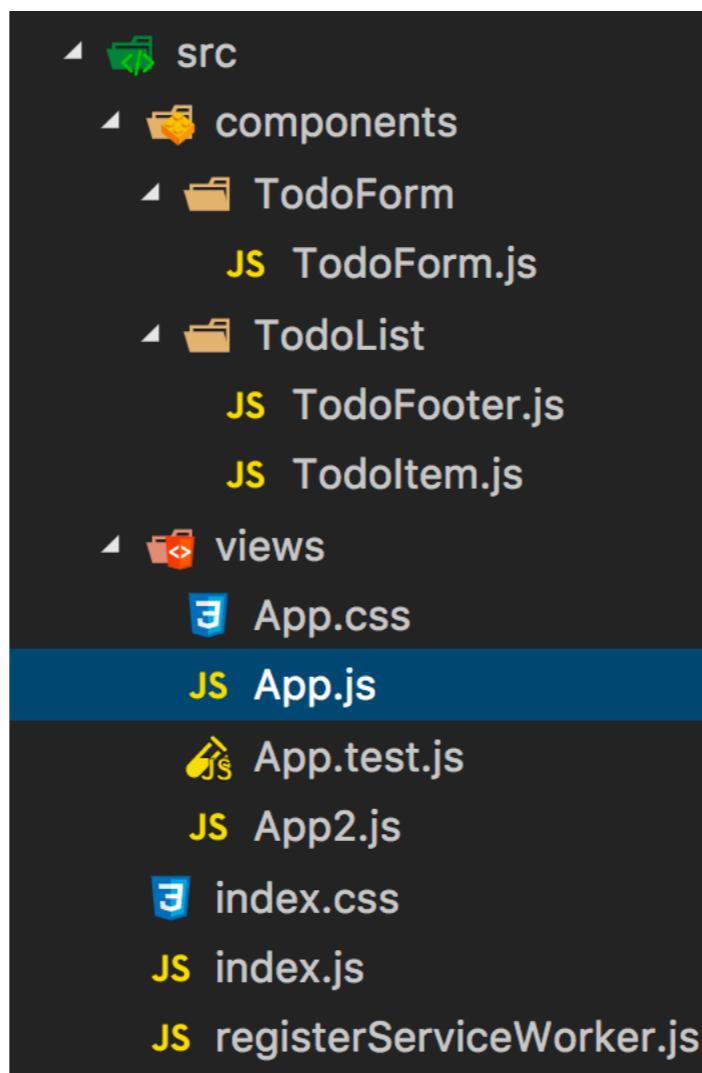
Let's Go!

Folder Structure

(todo-list-folder)

專案時間：將剛剛的todo list拆解

學會容易維護的folder sturcture和利用props和state傳遞變數



複製剛剛的todo-list，Let's GO !

redux學習

為何用Redux

Redux是一個小型庫，具有簡單，有限的API，旨在成為應用程序狀態的可預測容器。它以與還原功能類似的方式運行，即功能編程概念。



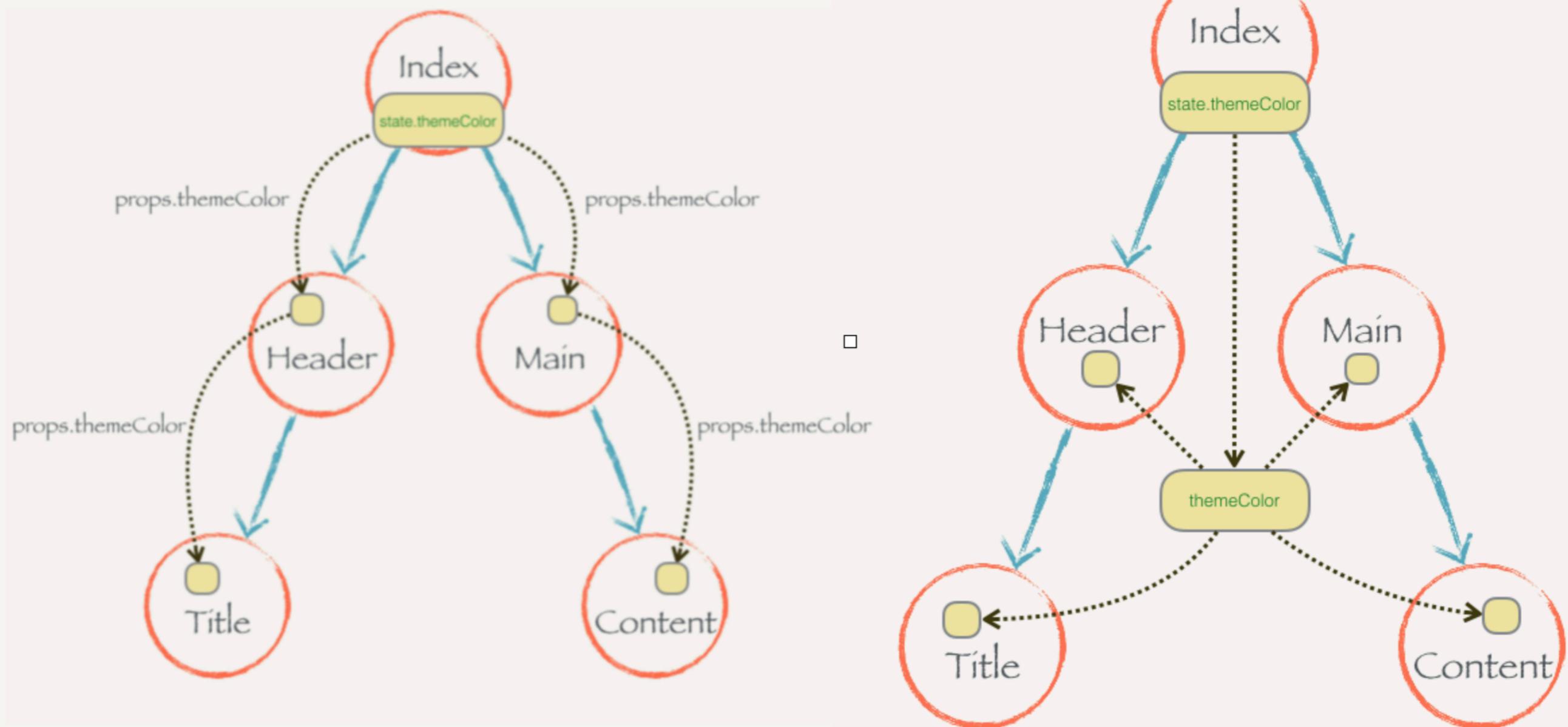
react 是單向數據流的。外部組件難以獲取內部組件的數據。



我們想要一個全域變數（全域的state）。

redux學習

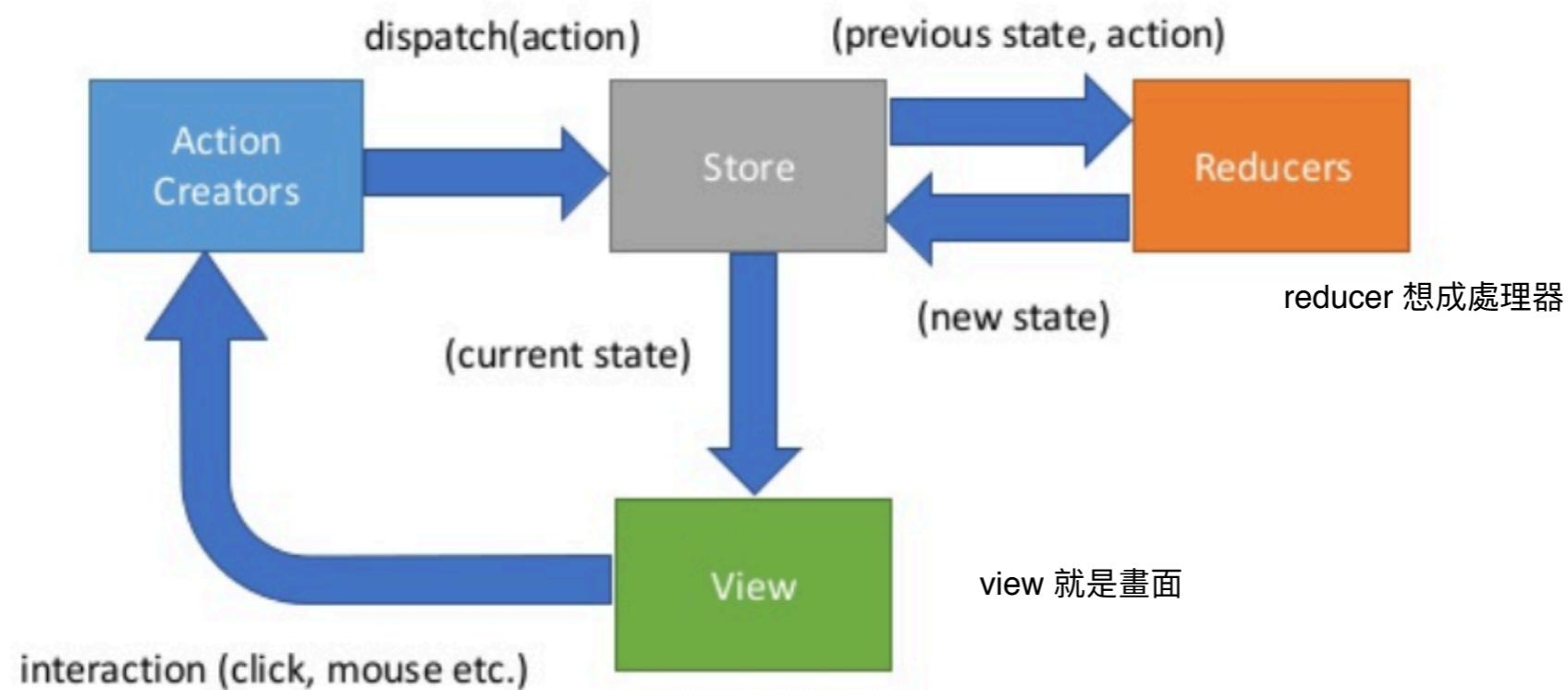
為何用Redux



redux學習

Redux Data Flow

why 要redux? 進階父子傳



redux-react-hook學習

加入redux步驟

npm install redux

npm install redux-react-hook

1.在src下加一個資料夾**redux**，裡面加一個檔案**store.js**

```
import { createStore } from 'redux'
```

放入createStore

```
//reducer
function names(state, action) {
  switch (action.type) {
    case 'ADD_NAME':
      return {...state, name:'親愛的'+action.payload.name}
    default:
      return state
  }
}
```

放入reducer這個function

(上一個state，要做的動作)

es6展開運算子

定所有需要的全域變數初數值

```
//store
let store = createStore(names,{name:''})
export default store;
```

執行createStore(reducer,初始化的state)

redux-react-hook學習

加入redux步驟

2. 在**index.js**中，加上

```
import {StoreContext} from 'redux-react-hook';
import store from './store';
```

並把原本的

```
ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

讓這隻index.js 都可以用到redux裡面的store.

改成

```
ReactDOM.render(
  <StoreContext.Provider value={store}>
    <App />
  </StoreContext.Provider>,
  document.getElementById('root')
);
```

redux-react-hook學習

加入redux步驟

3. 存入/讀取

```
import {useMappedState,useDispatch} from 'redux-react-hook';
```

```
...
```

```
//讀取 const name = useMappedState(state=> state.name);
```

在JSX中 <h1>{name}</h1> 的React Todo List

```
//存入
```

```
const dispatch = useDispatch();
```

```
<button onClick={
```

```
  ()=>{  
    dispatch({type:"ADD_NAME",payload:{name:'Kevin'}})  
  }  
}>進場</button>
```

這就是action!

如果在useEffect存入
只要dispatch({type:"ADD_NAME",payload:{name:'Kevin'}})

接下來只要在任何地方寫下讀取的程式碼，就能讀到全域的state了！

redux-react-hook學習

reducer&action

解析reducer&action

```
//action  
{type:'ADD_NAME',payload:{name:'kevin'}}
```

```
//reducer  
function names(state, action) {  
  switch (action.type) {  
    case 'ADD_NAME':  
      return {...state,name:'親愛的'+action.payload.name}  
    default:  
      return state  
  }  
}
```

假設原本
state={
 number:53
}

↓

state={
 number:53,
 name:'親愛的kevin'
}

(todo-list-folder-redux)

實作：todo list 進階版

將剛剛的專案先加入prompt詢問使用者
改成redux使用全域變數，把name放到header和input

The screenshot shows a browser window with the URL `localhost:3000`. A modal dialog box is displayed, asking for the user's name with the placeholder "請問你的名字是？". Below the dialog, the main application area displays the title "kevin的React Todo List". It lists two items: "跑步" (Run) and "喝水" (Drink Water), where "喝水" is checked. The status bar indicates "1未完成/2總數" (1 incomplete/2 total). At the bottom, there is a text input field containing "kevin快來增加任務吧..." and a button labeled "新增任務" (Add Task).

localhost:3000 顯示

請問你的名字是？

取消 確定

kevin的React Todo List

跑步

喝水

1未完成/2總數

kevin快來增加任務吧...

新增任務

useEffect 生命週期

(useeffect-practice)

```
useEffect(()=>{
  console.log('進入畫面，或有更新1')
})
```

```
useEffect(()=>{
  console.log('進入畫面，或有更新2')
  return (async()=>{
    setLogin('已登入')
    const dataFetch = await fetch('http://localhost:5050/')
    const data = await dataFetch.json()
    setData(data)
  })
}, [status])
```

讀後端api的方式

備註：接api，接firebase，都寫在useEffect裡面。

Firebase 教學

The screenshot shows a web browser window displaying the [ML Kit for Firebase](https://firebase.google.com/products/ml-kit/?authuser=0) page. The URL in the address bar is <https://firebase.google.com/products/ml-kit/?authuser=0>. The page features a navigation bar with links for Products, Use Cases, Pricing, Docs, and Support. A search bar is located on the right side of the navigation bar. A red circle highlights the "前往控制台" (Go to Console) button, which is positioned next to a user profile icon. Below the navigation bar, the main content area has a heading "ML Kit for Firebase" with a "BETA" badge, followed by the text "Machine learning for mobile developers". There are two buttons: "VISIT CONSOLE" and "VIEW DOCS". The bottom of the page shows a purple footer bar with the word "Mobile" partially visible.

Firebase 教學

The screenshot shows the Firebase console homepage. At the top, there is a navigation bar with a back arrow, forward arrow, refresh button, a lock icon indicating a secure connection to <https://console.firebaseio.google.com/u/0/>, and various browser icons. Below the navigation bar, the Firebase logo is on the left, and a user profile picture and a '取得說明文件' (Get Documentation) link are on the right.

歡迎使用 Firebase!

Google 提供多項工具，協助您開發優質應用程式、與使用者交流互動，以及透過行動廣告賺取更多收益。

[瞭解詳情](#) [說明文件](#) [支援](#)

近期專案

LoveLineSimpleUserData
lovelinesimpleuserdata

love testor
love-testor

A large graphic on the right side of the page features a smartphone displaying code snippets related to FIRAuth and payment processing, with orange cards showing padlocks and dollar signs.

新增專案 (The 'Add Project' button is highlighted with a red oval.)

探索示範專案 (Explore Sample Projects)

Firebase 教學

新增專案

⚠ 再增加 3 個專案就達到您的專案數量上限。

專案名稱 Android + iOS + </>

reactjs-practice

提示：專案可連結不同平台的多個應用程式 [?](#)

專案 ID [?](#)

reactjs-practice-f3ec5 

數據分析和計費區域 [?](#)

台灣

使用預設的 Google Analytics for Firebase 資料分享設定

✓ 提供 Analytics (分析) 資料給 Google，協助我們改善產品和服務
✓ 提供 Analytics (分析) 資料給 Google 以取得技術支援
✓ 提供 Analytics (分析) 資料給 Google 以啟用基準化功能
✓ 提供 Analytics (分析) 資料給 Google 帳戶專家

Firebase 教學

The screenshot shows the Firebase console interface for a project named "reactjs-practice". The left sidebar has sections for Development (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality (Crashlytics, Performance, Test Lab), and Data Analysis (Spark, free monthly \$0 USD). The main area is titled "Cloud Firestore" and describes it as "新一代 Realtime Database 擁有更強大的查詢和自動調整資源配置功能". A large orange button labeled "建立資料庫" is highlighted with a red oval. A modal window titled "Cloud Firestore 安全性規則" explains that rules must be defined after defining data structure. It offers two options: "以鎖定模式啟動" (selected) and "以測試模式啟動". The "allow read, write;" rule is shown in the code editor. A warning message states: "擁有您資料庫參考資料的所有使用者都能讀取或寫入您的資料庫". At the bottom, a note says: "啟用 Cloud Firestore 測試版後，您將無法在這個專案中使用 Cloud Datastore (特別是透過相關的 App Engine 應用程式)".

安全 | https://console.firebaseio.google.com/u/0/project/reactjs-practice-f3ec5/database

取得說明文件

reactjs-practice

測試版

Cloud Firestore

新一代 Realtime Database 擁有更強大的查詢和自動調整資源配置功能

建立資料庫

Cloud Firestore 安全性規則

定義資料結構之後，您必須撰寫規則來保護資料。[瞭解詳情](#)

以鎖定模式啟動
拒絕所有讀寫作業，保護資料庫的私密性

以測試模式啟動
允許對資料庫執行所有讀寫作業，加速完成設定程序。

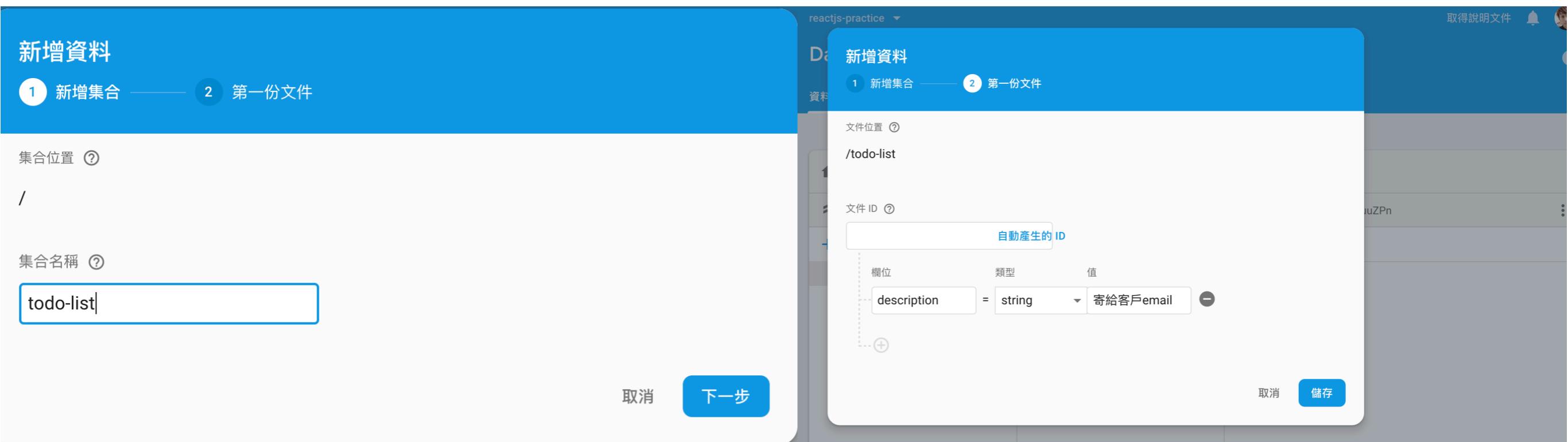
```
service cloud.firestore {  
  match /databases/{database}/documents {  
    match /{document=**} {  
      allow read, write;  
    }  
  }  
}
```

! 擁有您資料庫參考資料的所有使用者都能讀取或寫入您的資料庫

啟用 Cloud Firestore 測試版後，您將無法在這個專案中使用 Cloud Datastore (特別是透過相關的 App Engine 應用程式)。

取消 啟用

Firebase 教學



Firebase 教學

The screenshot shows the Firebase Project Overview page for a project named "reactjs-practice". The left sidebar includes sections for Authentication, Database, Storage, Hosting, Functions, and ML Kit. The main content area displays the project's details:

項目資訊	值
專案名稱	reactjs-practice
公開名稱	project-145776950324
專案 ID	reactjs-practice-f3ec5
網路 API 金鑰	AlzaSyDXwu6kVbJuzxyGLCCEQBGFLF9ivUZ4Vyl

在顶部菜单栏，"使用者和權限" 菜单项被高亮显示。右侧有一个悬停菜单，包含 "專案設定" 和 "使用者和權限" 选项。

Firebase 教學

在src底下新增fire.js

```
import firebase from 'firebase'

let config = {
  apiKey: "AIzaSyDXwu6kVbJuzxyGLCCEQBGF9ivUZ4VyI",
  authDomain: "reactjs-practice-f3ec5.firebaseio.com",
  databaseURL: "https://reactjs-practice-f3ec5.firebaseio.com",
  projectId: "reactjs-practice-f3ec5",
  storageBucket: "reactjs-practice-f3ec5.appspot.com",
  messagingSenderId: "145776950324"
};

let fire = firebase.initializeApp(config);

export default fire;
```

npm install firebase

```
import fire from './fire'

const db = fire.firestore();

const settings = {timestampsInSnapshots: true};

db.settings(settings);
```

將 Firebase 加入您的網路應用程式

複製下方的程式碼片段，然後貼到 HTML 底端 (其他 script 標記前)。

```
<script src="https://www.gstatic.com/firebasejs/5.2.0.firebaseio.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyDXwu6kVbJuzxyGLCCEQBGF9ivUZ4VyI",
    authDomain: "reactjs-practice-f3ec5.firebaseio.com",
    databaseURL: "https://reactjs-practice-f3ec5.firebaseio.com",
    projectId: "reactjs-practice-f3ec5",
    storageBucket: "reactjs-practice-f3ec5.appspot.com",
    messagingSenderId: "145776950324"
  };
  firebase.initializeApp(config);
</script>
```

Get Started with Firebase for Web Apps [\[link\]](#)
Firebase Web SDK API Reference [\[link\]](#)
Firebase Web Samples [\[link\]](#)

在想使用firebase的檔案中

Firebase 教學

存入

Set document

```
let teamRef = db.collection('todo-list').doc('test').set({  
  name:'hello',  
  email:'neversaynever0502@gmail.com'  
})
```

Add document

```
var addDoc = db.collection('cities').add({  
  name: 'Tokyo',  
  country: 'Japan'  
}).then(ref => {  
  console.log('Added document with ID: ', ref.id);  
});
```

Update document

```
db.collection('todo-list').doc(ref.id).update({  
  key:ref.id,  
})
```

Firebase 教學

讀取doc一次

```
let teamRef = db.collection('todo-list').doc('XXX')
teamRef.get().then((doc) => {
  if (doc.exists) {
    let docData = doc.data();
  }
})
```

重複監聽

```
let Ref = db.collection('todo-list').doc('XXX');
Ref.onSnapshot((doc) => {
  if (doc.exists) {
    let docData = doc.data();
  } else {
    console.log('讀取失敗')
  }
})
```

Firebase 教學

Get all documents in a collection(一次)

```
var citiesRef = db.collection('cities');
var allCities = citiesRef.get()
  .then(snapshot => {
    snapshot.forEach(doc => {
      console.log(doc.id, '=>', doc.data());
    });
  })
  .catch(err => {
    console.log('Error getting documents', err);
 });
```



EO0nJoDxGvm3PdEtyfFI =>

{description: "寄發email給客戶", done: false, key: "EO0nJoDxGvm3PdEtyfFI"}

O5p4F7W77LarvQfmIHUD =>

{description: "跑步", done: false, key: "O5p4F7W77LarvQfmIHUD"}

Get all documents in a collection (重複監聽)

```
let Ref = db.collection('messages');
Ref.onSnapshot((snapshot) => {
  snapshot.docChanges().forEach(change => {
    if(change.type=='added'){
      console.log(change.doc.id, '=>', change.doc.data());
    }
  });
});
```

Get all documents in a collection + 排序

```
let Ref = db.collection('messages').orderBy('timestamp');
Ref.onSnapshot((snapshot) => {
  snapshot.docChanges().forEach(change => {
    if(change.type=='added'){
      console.log(change.doc.id, '=>',
        change.doc.data());
    }
  });
});
```

專案時間：

todo-list進階版（firebase+todo-list）

重新整理將會永遠保存資料

React Todo List

- 寄發email給客戶
- 跑步
- 練習吉他
- 做文件
- 寫react.js

2未完成/5總數

 新增任務

複製剛剛的資料夾，Let's GO !

專案實作：聊天室

react+firebase

The diagram illustrates a wireframe for a messaging application. It features a large rectangular container with a double-line border. Inside, there are four horizontal rows, each consisting of two input fields and a send button. The first three rows are identical, showing a '人名' (Name) field and a larger '文字' (Text) field. The fourth row at the bottom is partially cut off on the right. Between the second and third rows, there is a small ellipsis '...' indicating more messages. At the very bottom of the interface, there is a footer bar with three input fields: '輸入人名' (Input Name), '輸入文字' (Input Text), and a blue '送' (Send) button.

人名	文字	
人名	文字	
...		
輸入人名	輸入文字	送

專案實作：聊天室

去firebase新增資料格式

The screenshot displays the MongoDB Compass interface, illustrating a database structure for a chatroom application. The top navigation bar shows the path: Home > chatroom > chatroom01.

The main view shows a collection named "chatroom01" with the following structure:

- chatroom01**: A document with fields:
 - name**: "chatroom01"
 - messages**: An array of message documents.
- messages**: An array of message documents, each with fields:
 - messageText**: "哈囉"
 - timestamp**: 2018年7月19日 下午12:00:23 [UTC+8]
 - username**: "小明"

A modal window is open, showing the detailed structure of one message document:

- 1eSGDyoEwhjwqiJApq6q**: A document with fields:
 - messageText**: "哈囉"
 - timestamp**: 2018年7月19日 下午12:00:23 [UTC+8]
 - username**: "小明"

程式碼Let's Go

人生就像滾雪球，你只要找到濕的雪，和很長的坡道，雪球就會越滾越大。 – 巴菲特

恭喜完成～

可以開兩個視窗互相測試即時傳訊

A screenshot of a web browser window titled "localhost:3000". The browser interface includes a toolbar with various icons, a menu bar with "檔案" and "檢視" options, and a sidebar with links to "應用程式", "常用社交與搜尋", "google 相關", "javascript學習", "論文", "html 工具", "[解除]", "ptt.cc", "機器學習", "英文學習", "股票期貨選擇權", "設計", "動漫電影", "雲端", and "其他書籤". The main content area displays a conversation between two users:

小明: 來玩玩~~~

小明: 小皮球

Kai: 香蕉油

Kai: 滿地開花21

小明: 別再亂接了，我們只是測試人員

小明: 負責測試這是不是好的聊天室

Kai: 沒錯，現在看起來好像沒什麼問題

小明: 很棒！我終於用react+firebase串起聊天室來了

Kai: 恭喜你~~~~雖然css還有很多可以改的，但功能已經ok了！！！

小明: 恩恩

小明: 很棒

Kai: 讓我們一起成長吧！！！

The bottom of the screen shows a message input field with "Kai" and "輸入訊息..." placeholder text, and a blue "送出" (Send) button.

(chatroom-pro)
專案實作：聊天室進階版
react+firebase

請輸入房間號碼....

進入

專案實作：聊天室進階版

react+firebase

房間號碼

人名 文

人名 文

...

輸入人名 輸入文字 送

恭喜完成～

可以開兩個視窗互相測試即時傳訊

房間編號: chatroom01

離開房間

小明: 很棒！我終於用react+firebase串起聊天室來了

Kai: 恭喜你~~~~~雖然css還有很多可以改的，但功能已經ok了！！！

小明: 恩恩

小明: 很棒

Kai: 讓我們一起成長吧！！！

小明: 你好哇

小明: 哈囉囉

小王

輸入訊息...

送出

輸入房間編號: chatroom02

進入

房間編號: chatroom02

離開房間

小明: 哈囉你好

小美: 哈囉這是第二間房間

小美: 挖到第二間房間了！！！！！

小美: 如果愛會說話~~~~~

小王: 如果風愛上沙~~~~~

小王: 如果，有些是遺忘在每個地方~~~~~

小美: yahoo~~~~~

小王

輸入訊息...

送出

免費部署上架教學

使用firebase hosting

The screenshot shows the Firebase console interface. On the left, there's a sidebar with various services: Authentication, Database, Storage, **Hosting** (which is circled in red), Functions, ML Kit, Crashlytics, Performance, Test Lab, Dashboard, Events, Conversions, and Spark. The main area is titled "Hosting" and has a blue header bar with "設定代管服務" (Setup Hosting Service). It shows two steps: "1 安裝" (Step 1: Install) and "2 部署" (Step 2: Deploy). Below the steps, it says "開啟終端機視窗, 然後進入到您的網站目錄, 或新建一個網站目錄" (Open terminal window, then enter your website directory, or create a new website directory). It provides three command-line snippets: "登入 Google: \$ firebase login", "啟動專案: \$ firebase init", and "部署網站: \$ firebase deploy". A blue "完成" (Finish) button is at the bottom right.

```
Kai-Linde-MacBook-Pro:chatroom-pro kai-lin$ firebase login
```

```
Update available: 3.18.5 (current: 3.18.4)  
Run npm install -g firebase-tools to update.
```

```
Already logged in as neversaynever0502@gmail.com  
Kai-Linde-MacBook-Pro:chatroom-pro kai-lin$ firebase init
```

0.npm run build => 生產build資料夾

1.安裝Firebase cli

(npm install -g firebase-tools)

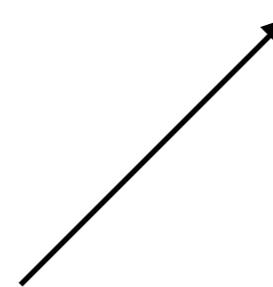
2.

firebase login

firebase init

選擇「Hosting」

按「空白鍵」選擇，再按enter



```
You're about to initialize a Firebase project in this directory:
```

```
/Users/kai-lin/專案/reactjs課程-入門實做課/測試/chatroom-pro
```

```
? Which Firebase CLI features do you want to setup for this folder? Press Space to  
select features, then Enter to confirm your choices.  
o Database: Deploy Firebase Realtime Database Rules  
o Firestore: Deploy rules and create indexes for Firestore  
o Functions: Configure and deploy Cloud Functions  
x o Hosting: Configure and deploy Firebase Hosting sites  
o Storage: Deploy Cloud Storage security rules
```

```
? Select a default Firebase project for this directory:
```

```
slave game (slave-game)  
CodeforgenderHackthon (codeforgenderhackthon)  
Coconut Grove Chat (coconut-grove-chat)  
x > reactjs-practice (reactjs-practice-f3ec5)  
heyWebDate (heywebdate)  
love testor (love-testor)  
cxinno (cxinno-tw)
```



選擇你要的「專案」

```
Your public directory is the folder (relative to your project directory) that will contain Hosting assets to be uploaded with firebase deploy. If you have a build process for your assets, use your build's output directory.
```

```
? What do you want to use as your public directory? (public) build
```

輸入「build」為你的公開資料夾

```
? What do you want to use as your public directory? build  
? Configure as a single-page app (rewrite all urls to /index.html)? Yes  
? File build/index.html already exists. Overwrite? (y/N) N
```

輸入「Y」為你只有一個html檔

輸入「N」不要覆蓋！

```
? File build/index.html already exists. Overwrite? No  
i Skipping write of build/index.html  
  
i Writing configuration info to firebase.json...  
i Writing project information to .firebaserc...  
  
✓ Firebase initialization complete!
```

設定完成，接下來輸入「firebase deploy」

```
==== Deploying to 'reactjs-practice-f3ec5'...  
i  deploying hosting  
i  hosting: preparing build directory for upload...  
✓  hosting: 10 files uploaded successfully  
  
✓  Deploy complete!  
  
Project Console: https://console.firebaseio.google.com/project/reactjs-practice-f3ec5/overview  
Hosting URL: https://reactjs-practice-f3ec5.firebaseio.com
```

恭喜上傳成功，網址為：<https://reactjs-practice-f3ec5.firebaseio.com/>



Thanks For Listening

Q&A 時間

完整程式碼上完課後可跟老師索取