

React + Webpack

milkmidi
2020

環境設定 1

install nodejs(10.x以上)

```
安裝完後，開啟 terminal 輸入  
node -v  
看有沒有出現版本號
```


環境設定 2

install vscode

vscode extensions

- 1 Live Server
- 2 JavaScript (ES6) code snippets
- 3 ES7 React/Redux/GraphQL/React-Native snippets
- 4 Highlight Matching Tag
- 5 Path Intellisense
- 6 REST Client
- 7 Todo Tree
- 8 Guides

環境設定 3

`vscode jsx emmet`

環境設定 3

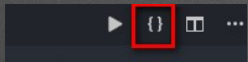
開啟使用者設定

vscode > Preference > User Settings

或按快速鍵啟動：

Windows：Ctrl + ,

Mac：Command + ,

接著再按vscode右上角的  {} 圖示，切換到 json 格式

```
{
  "emmet.includeLanguages": {
    "javascript": "javascriptreact"
  },
  "files.associations": {
    "*.js": "javascriptreact"
  }
}
```

熟悉 ES6

可先參考奶綠我寫的這一篇 [React 起手式 ES6篇](#)

01_helloworld

將學會

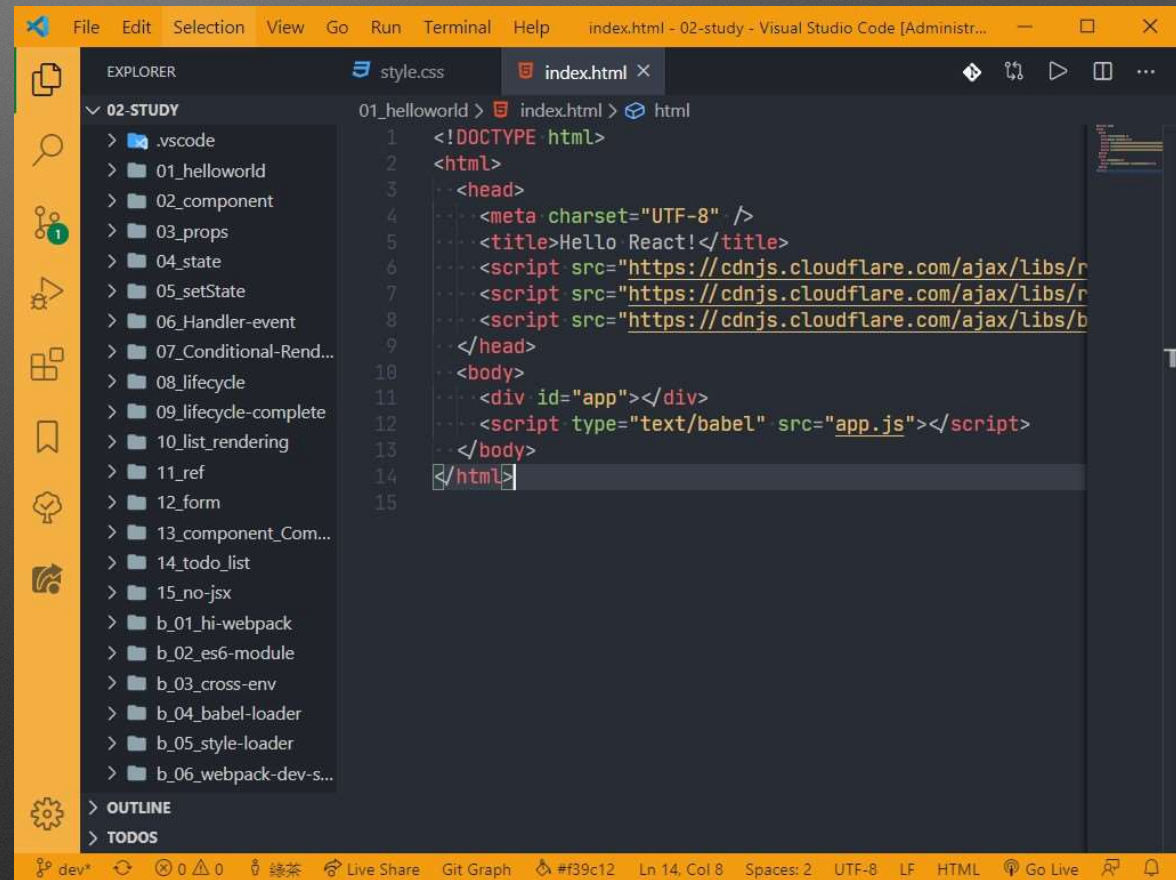
0 開啟專案資料夾

1 啟動 live server

01_helloworld

0 開啟專案資料夾

vscode / File / Open Folder
選擇老師分享的資料夾 02-study
完成後會如右邊

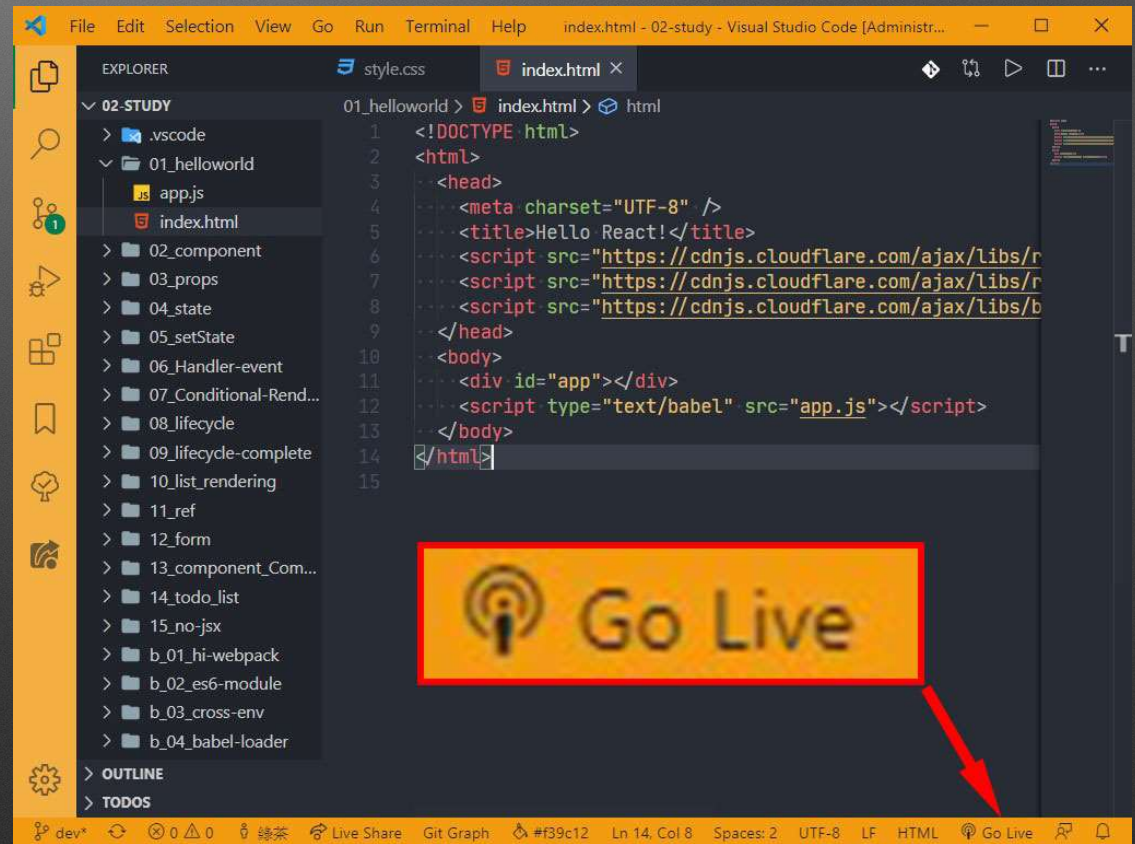


01_helloworld

1 啟動 live server

開啟 01_helloworld/index.html (一定要是html)

vscode 右下角有個 Go Live 按下去



02_component

將學會

0 React.Component

1 Functional Component

03_props

將學會

0 Component 怎麼接參數

1 jsx emmet

03_props

1 jsx emmet

.card 按 tab 鍵

```
<div className="card"></div>
```

.card>.img+.name 按 tab 鍵

```
<div className="card">  
  <div className="img"></div>  
  <div className="name"></div>  
</div>
```

.card>ul>(li>a.index\$)*2

```
<div className="card">  
  <ul>  
    <li><a href="" className="index1"></a></li>  
    <li><a href="" className="index2"></a></li>  
  </ul>  
</div>
```


04_state

將學會

0 `React.Component` state

1 `Functional Component` 沒有 state, 沒有生命週期

考

放兩個 `Counter Component`

一個 `state.count` 為 0

另一個 `state.count` 為 1

05_setState

將學會

React.Component 更新 state

setState 有兩種寫法：

```
1 this.setState({});
```

```
2 this.setState(function(prevState){  
    return {}  
});
```


06_Handler-event

將學會

React.Component EventHandler寫法

07_Conditional-Rendering

將學會

依變數來決定要不要 Render DOM

08_lifecycle-basic

將學會

`React.Component` 生命週期

其實只要記這兩個就夠用了

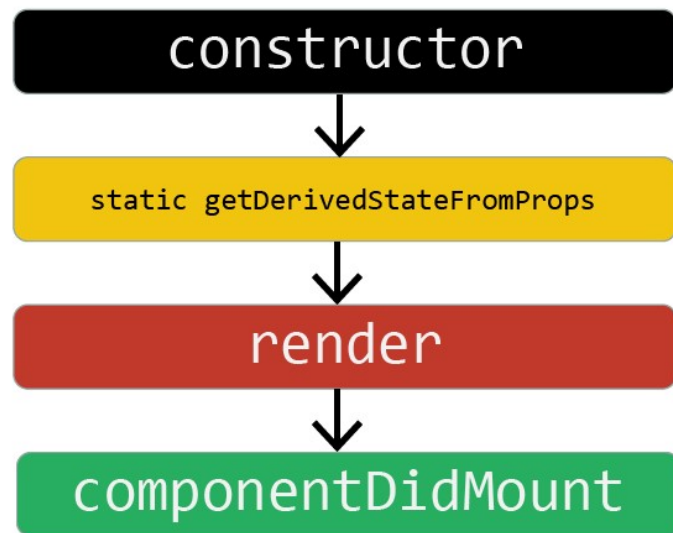
`componentDidMount`

`componentWillUnmount`

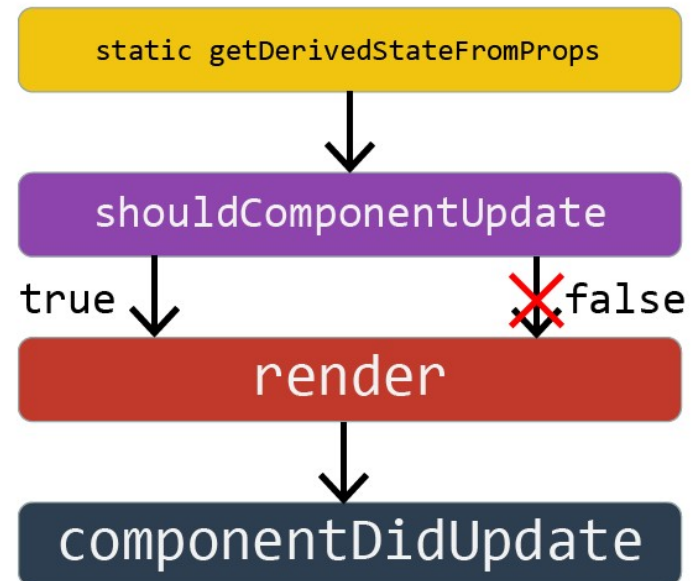
[官方文件請點我](#)

09_lifecycle-complete

Mounting



Updating



10_list-rendering

將學會
使用回圈來產生 DOM

```
key 是重點，一定要加  
this.state.list.map(function (text) {  
  return <li key={text}>{text}</li>;  
})
```

11_ref

將學會

- 0 找到 `React.Component` 裡指定的 DOM 元素
(不使用 `document.querySelector`)
- 1 `setState` 是非同步函式

12_form

將學會

0 各種表單 `input` 應用

1 `ref` 可以取得 DOM 之外，也可以取得 `class` 實體。

13_component-Communication

將學會
父組件與子組件的溝通(限定一等親)

14_todo-list

將學會

TodoList 開發

共分三個元件

TodoForm.js - 輸入框，可以新增 Todo 文字

TodoItem.js - 顯示目前的 Todo 列表資料，點擊可切換

TodoList.js - 資料存放在這

15_no-jsx

將學會

0 不寫 `jsx`，直接寫 `js`（應該不會有人這麼做啦）

1 `class properties` 轉換原理



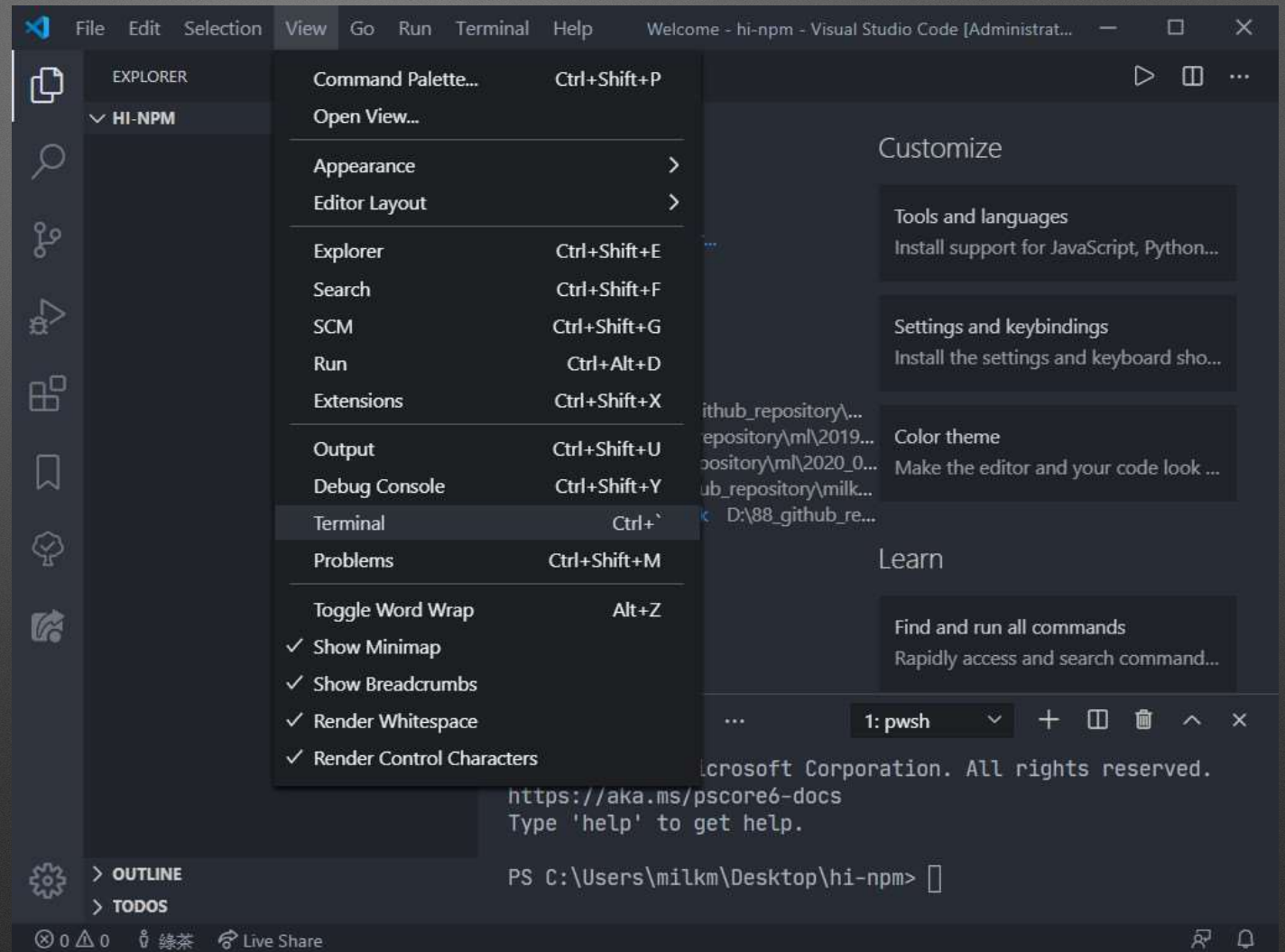
hi npm

- 0 新增一個空白的資料夾
- 1 用 `vscode` 開啟該資料夾
- 2 啟動 `vscode Terminal`
`vscode /View / Terminal`

hi npm

啟動 Terminal 後
路徑會自動指向目前的資料夾

在 Terminal 輸入
`npm init`
然後一直下一步即可
就會自動產生 `package.json`



安裝新的 package

`npm install` 套件包 (或是用縮寫 `npm i` 套件包)

例：`npm i jquery`

(所有的 package 套件包都是小寫)

安裝完後，會自動在你的 `package.json dependencies`：寫入

安裝新的 package

補充：

`npm i -D 套件包`

會把套件寫到 `devDependencies`：

一般會加 `-D` 都是用在有跑自動測試的專案上

`npm i --production`

(only install dependencies packages)



hi_webpack

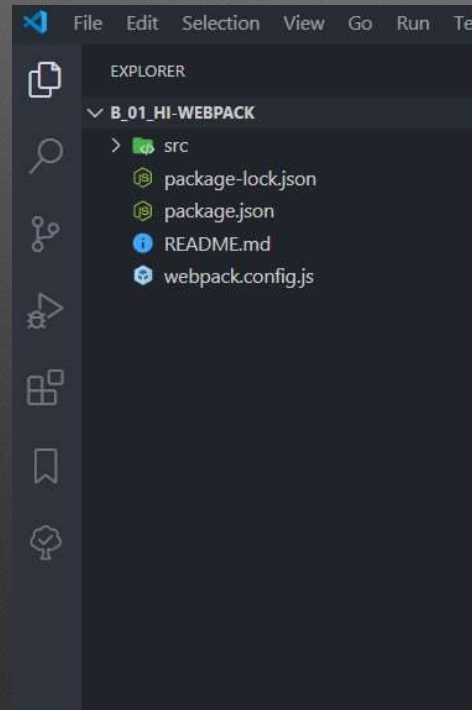
將學會

0 開啟專案的方式

1 重新安裝 package

hi_webpack

之後所有的專案開啟方式
都要改成開啟該資料夾
專案根目錄要有 `package.json`才是正確的



package.json

scripts: 可透過 `npm run scriptName` 來啟動。

如下圖

`npm run start` (可縮打成 `npm start`, 本機開發用)

`npm run build` (打包成 production 版, 可丟上正式機)

devDependencies:

dependencies:

這個專案有用到的套件包和版本號。

```
{
  "name": "milkmidi-webpack-example",
  "version": "1.0.0",
  "description": "webpack example",
  "author": "milkmidi",
  "scripts": {
    "start": "webpack --mode development",
    "build": "webpack --mode production && node dist/bundle.js"
  },
  "devDependencies": {
    "webpack": "^4.39.1",
    "webpack-cli": "^3.3.6"
  }
}
```


重新安裝所需的 package

```
npm i
```

node_modules

npm i 重新安裝 packages 自動產生
檔案很大，本機開發時才會用到
一般都會 gitignore 掉。

怎麼刪 node_modules

Mac：直接按 delete

PC：直接按 delete（刪超慢的）

優化刪法：

先安裝 global package

npm i -g rimraf

接著就可以用 rimraf ./node_modules 來刪，會快很多

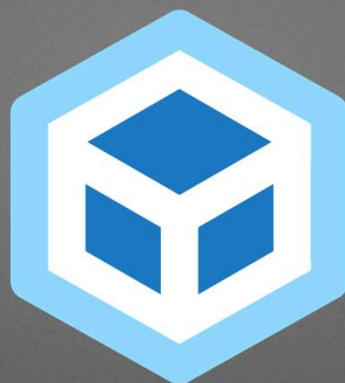


b_02_es6-module

將學會

0 es6 module (import export)

1 commonjs module (require)



b_03_cross-env

將學會

0 改用 cross-env 來設定 NODE_ENV 變數

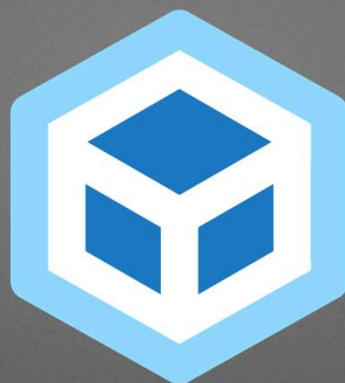
process.env.NODE_ENV 會依據 cross-env 的設定而有所不同

process.env.NODE_ENV 只能是 development / production 兩種



考

如果現在有測試機，stage機，production機
分別要吃不同的參數該怎麼辦？



b_04_babel-loader

將學會

0 babelrc 設定



b_05_style-loader

將學會

0 style-loader

1 url-loader



The image shows the Path Intellisense extension interface in Visual Studio Code. On the left is a circular icon with a red dot and a red slash. To the right of the icon, the text 'Path Intellisense' is displayed in a large font, followed by the identifier 'christian-kohler.path-intellisense' in a smaller font. Below this, the author 'Christian Kohler' is listed, followed by a download icon and the number '2,711,848'. To the right of the download count are five yellow stars. Further right are links for 'Repository' and 'License'. Below these links, the description 'Visual Studio Code plugin that autocompletes filenames' is shown. At the bottom, there are two buttons: 'Disable' with a downward arrow and 'Uninstall'. To the right of these buttons, a message states 'This extension is enabled globally.'

Path Intellisense christian-kohler.path-intellisense

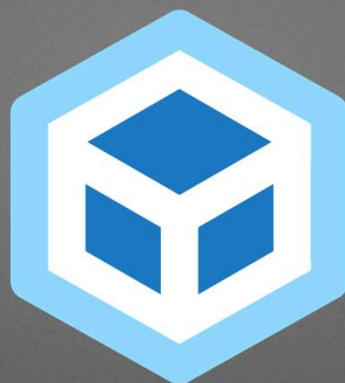
Christian Kohler | 2,711,848 | ★★★★★ | Repository | License

Visual Studio Code plugin that autocompletes filenames

[Disable ▼](#) [Uninstall](#) This extension is enabled globally.

安裝完成後，進 vscode settings

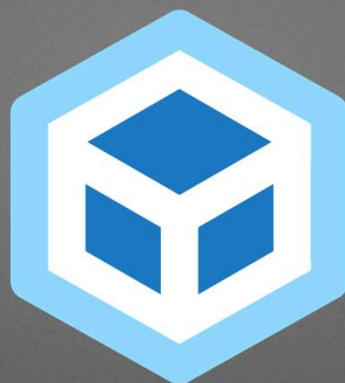
```
{
  "path-intellisense.mappings": {
    "~img": "${workspaceRoot}/src/img/"
  },
}
```

b_06_webpack-dev-server

將學會

0 本機開發用 webpack-dev-server



b_07_mini-css-extract-plugin

將學會

0 如何將 css 檔存成實體的檔案

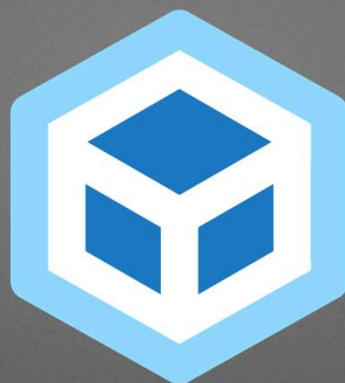


hash

hash : 每次打包都會產生一組新的 hash code(圖片用)

chunkhash : 有更新到的 js , hash code 才會不一樣

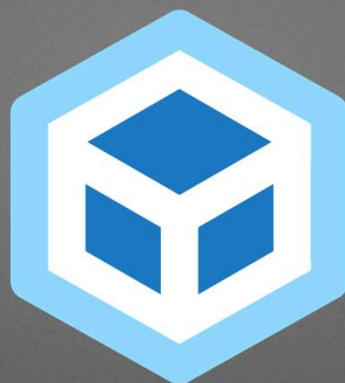
contenthash : js 沒更新 , 但 style 有更新 (css用)



b_08_DefinePlugin

將學會

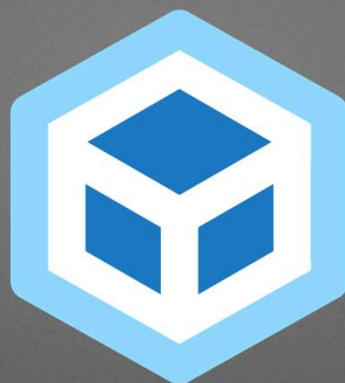
0 由 webpack 自定系統變數



b_09_optimization

將學會

0 JS 拆檔最佳化



b_10_multi-entry

將學會
0 多程式進入點



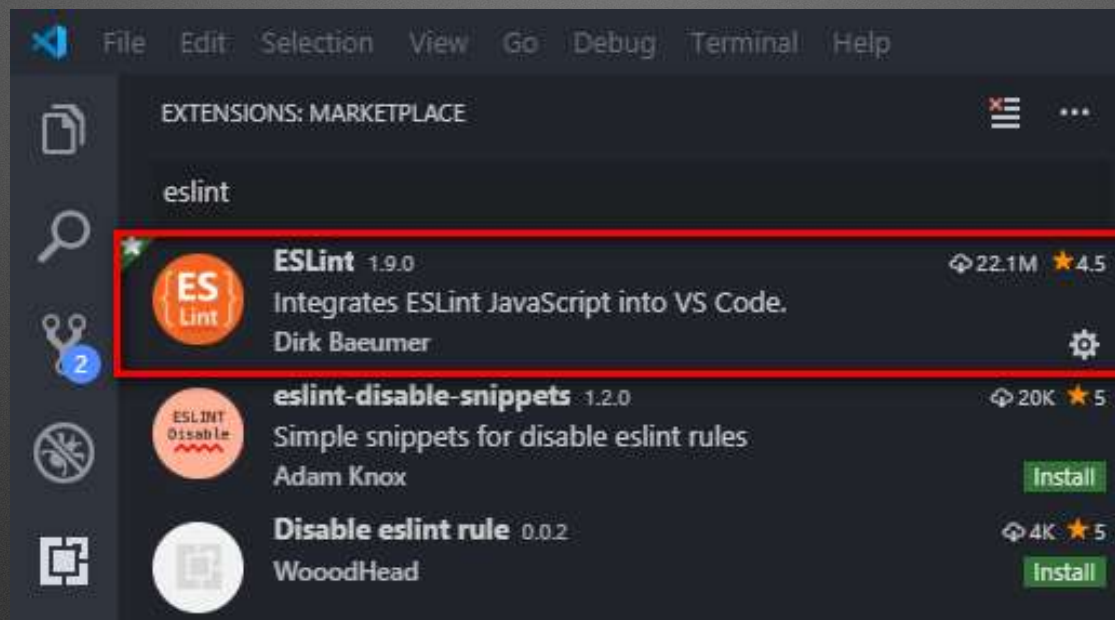
b_11_eslint

前端必學工具，太重要了

eslint

<https://eslint.org>

vscode extension



install eslint package

```
npm i -D eslint
```

react

```
npm i -D  
  eslint-config-airbnb  
  eslint-plugin-import  
  eslint-plugin-jsx-a11y  
  eslint-plugin-react
```

vscode settings

```
{  
  "editor.codeActionsOnSave": {  
    "source.fixAll.eslint": true  
  }  
}
```


eslint

.eslintrc.js

```
module.exports = {  
  extends: ['airbnb'],  
  env: {  
    browser: true,  
  },  
  parser: 'babel-eslint',  
  rules: {  
    'react/jsx-filename-extension': 0,  
    'no-param-reassign': ['error', { props: false }],  
    'react/prop-types': 0,  
    'jsx-a11y/click-events-have-key-events': 0,  
    'jsx-a11y/no-noninteractive-element-interactions': 0,  
    'react/button-has-type': 0,  
  },  
};
```

eslint

.eslintignore

不需要檢查的檔

```
build  
dist  
node_modules  
public  
!.eslintrc.js
```


eslint

ignore line

```
var a = 'milkmidi';    // eslint-disable-line
```

ignore next line

```
// eslint-disable-next-line  
var a = 'milkmidi';
```

block ignore

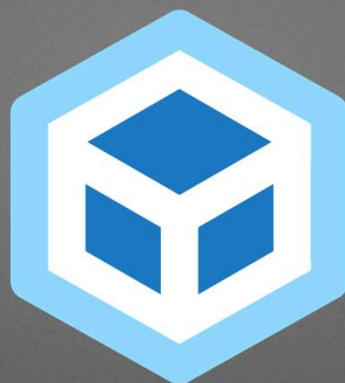
```
/* eslint-disable */  
var a = 'milkmidi';  
var b = 'react';  
/* eslint-enable */
```



b_12_remoting-debug-weinre

將學會

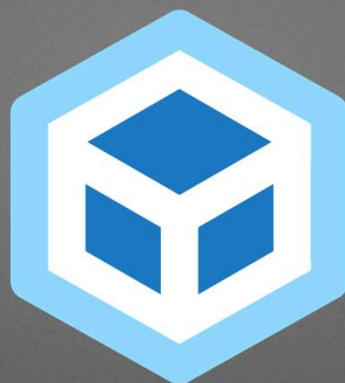
0 遠端手機 Debug
需要啟動兩個 terminal
npm run weinre
npm run start:weinre



b_13_flowjs

將學會

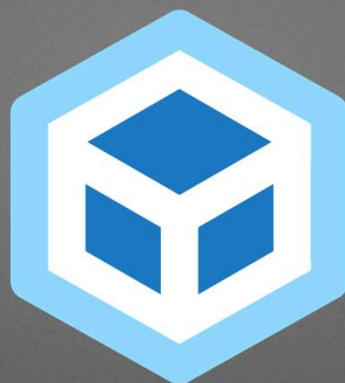
0 好用的 js 型別提式



b_14_babel-plugin-transform-remove-console

將學會

0 發佈 production 版時，自動移掉所有的 `console.log`



b_15_webpack-react

將學會

0 webpack react 設定

c_01_todo_list

將學會

0 TodoList module 化寫法

1 Chrome react devtool

c_01_todo_list_homework_delete_todo 考

實作 Delete 功能

TodoList.js

<input type="checkbox"/> 學會React	X
<input type="checkbox"/> 學會Webpack	X
<input type="checkbox"/> 年薪百萬	X

c_02_todo_list_api

將學會
接 api 與本機跨網域問題

c_03_taiwan_city_area_select

將學會
台灣縣市地區下拉選單用應

考

- 1 不要預設在基隆市
- 2 請加入請選擇
- 3 選了縣市地區後，請選譯的選項要拿掉

c_04_Promise

波動拳寫法，傳說中的 callback hell

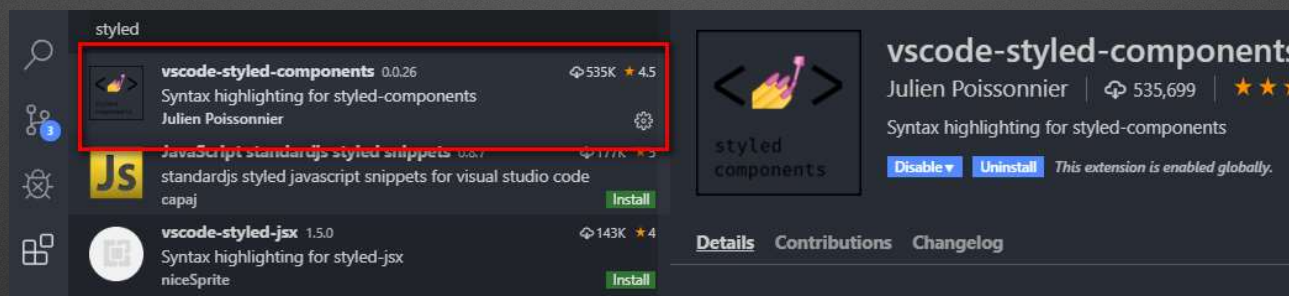


```
1 function hell(win) {
2   // for listener purpose
3   return function() {
4     loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
5       loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
6         loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
7           loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
8             loadLink(win, REMOTE_SRC+'/lib/underscore.min.js', function() {
9               loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
10                loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
11                  loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
12                    loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
13                      async.eachSeries(SERIALS, function(src, callback) {
14                        loadScript(win, BASE_URL+src, callback);
15                      });
16                    });
17                  });
18                });
19              });
20            });
21          });
22        });
23      });
24    });
25  });
26 }
```

c_05_styled-components

將學會
在 js 裡寫 style

vscode-styled-components



c_05_styled-components

src/components/Example01.js

```
import React from 'react';
import styled from 'styled-components';

const Title = styled.h1`
  font-size: 1.5em;
  text-align: center;
  color: blue;
`;

const Wrapper = styled.section`
  padding: 4em;
  background: grey;
`;

const Basic = () => (
  <Wrapper>
    <Title>hi</Title>
  </Wrapper>
);

export default Basic;
```



```
<html>
  <head>...</head>
  <body>
    <div id="root">
      <div class="app_container">
        <section class="sc-bwzfXH kMtIys">
          <h1 class="sc-bdVaJa bpnvxE">hi</h1> == $0
        </section>
      </div>
    </div>
    <script type="text/javascript" src="vendor-chunk.js"></script>
    <script type="text/javascript" src="app.js"></script>
  </body>
</html>
```


考

<input type="checkbox"/> Hello React	NORMAL
<input type="checkbox"/> Hello React	HOVER
<input checked="" type="checkbox"/> Hello React	CHECKED
<input type="checkbox"/> Hello React	DISABLED

- 1 用 styled-component 製作一個 Checkbox
- 2 檔名叫 : Checkbox.js
- 3 Props 有 {
 value: string, 選取後的值
 onChange: Function
 disabled: Boolean
 text: string 顯示用的文字(上圖的 Hello React 文字)
}
- 4 色碼 : #4094d8

d_01_react-router

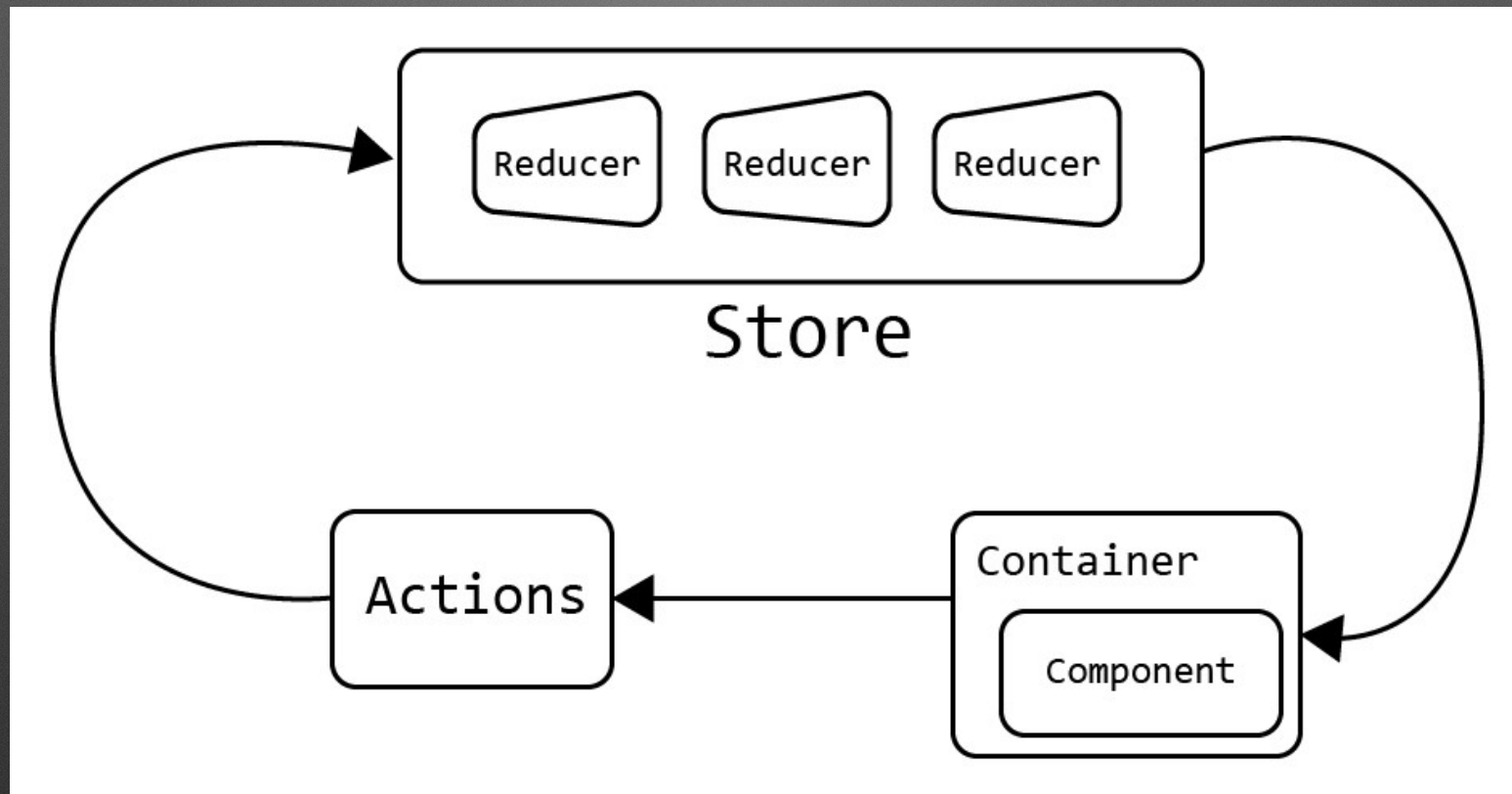
d_02_react-router-react-snap

<https://github.com/stereobooster/react-snap>

e_01_react-redux

<https://redux.js.org/>

redux-flow



redux evtool chrome

e_01_react-redux

src/index.js

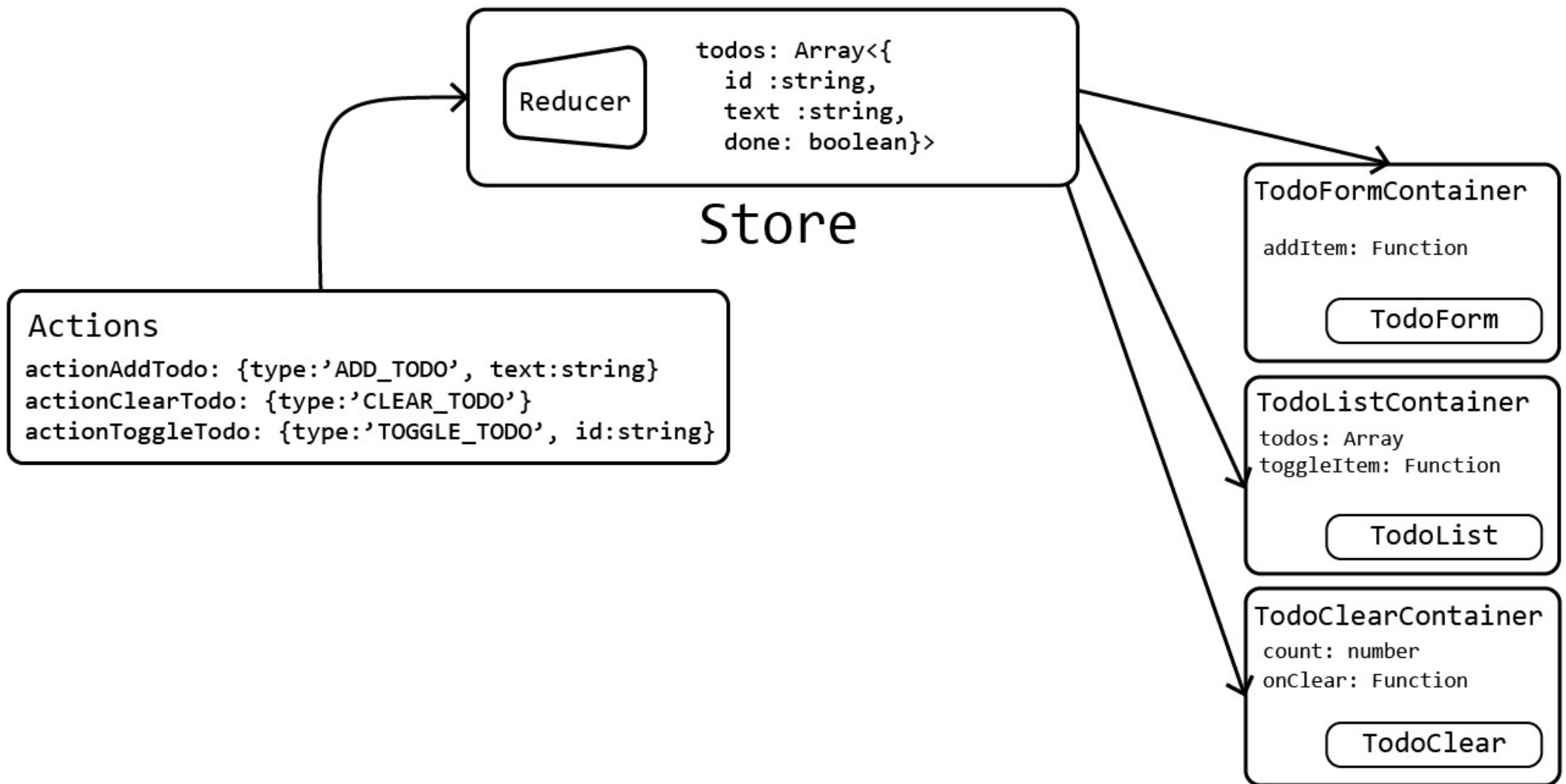
```
import React from 'react';
import ReactDOM from 'react-dom';
import { createStore } from 'redux';
import { Provider } from 'react-redux';
import reducer from './reducers';
import 'css/app.scss';

let preloadedState = null;
if (process.env.NODE_ENV === 'development') {
  preloadedState = window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__();
}
const store = createStore(reducer, preloadedState);

//const store = createStore(reducer);
```

略

e_02_react-redux-todo_list



e_03_react-redux-todo-list-api

f_01_PureComponent

直接看老師表演

Component

只要 `setState` 被呼叫，一定會觸發 `render`
除非自行加寫 `shouldComponentUpdate`

PureComponent

`setState` 被呼叫時
會執行 `shouldComponentUpdate` 並用 `shallowEqual` 判斷要不要重新 `render`

React.memo

<https://reactjs.org/docs/react-api.html#reactmemo>

React.memo is the equivalent to the
class version React.PureComponent

f_02_Optimizing-todo_list

f_03_higher-order-component

g_01_hooks

<https://reactjs.org/docs/hooks-intro.html>

React.useLayoutEffect

<https://reactjs.org/docs/hooks-reference.html#uselayouteffect>

用法和 `useEffect` 一樣
會在 DOM 更新完，畫面 render 之前

g_02_react-redux-hooks

將學會

redux hooks

g_03_react-router-redux-auth

將學會

router 整合 redux

限定要登入後才能進入該面頁

h_01_storybook

<https://storybook.js.org/>

plopjs

讓你三秒鐘產生一個 Component

<https://plopjs.com/>

h_02_testing-library

效能優化總結

React.memo

```
// Bad 🗨️  
每次都重新 render  
export default function StatelessComponent(props){  
  return (  
    <div>{props.value}</div>  
  )  
});  
  
// Good 😊  
// stateless 版的 PureComponent, 當 props 有更新時才會重新 render  
export default React.memo((props) => {  
  return (  
    <div>{props.value}</div>  
  )  
});
```


React.useMemo

```
// Bad 🗨️  
function Component(props) {  
  const someProp = heavyCalculation(props.item);  
  return <AnotherComponent someProp={someProp} />  
}
```

```
// Good 😊  
// 只有 props.item 改變時 someProp 的值才會被重新計算  
function Component(props) {  
  const someProp = useMemo(() => heavyCalculation(props.item), [props.item]);  
  return <AnotherComponent someProp={someProp} />  
}
```

React.PureComponent, shouldComponentUpdate

```
// Good 😊  
class AnotherComponent extends React.PureComponent {  
  render() {  
    return <div>{this.props.someOtherProp}</div>  
  }  
}  
  
// Good 😊  
class AnotherComponent extends React.Component {  
  shouldComponentUpdate(nextProps) {  
    return this.props.someOtherProp !== nextProps.someOtherProp;  
  }  
  render() {  
    return <div>{this.props.someOtherProp}</div>  
  }  
}
```


Props

```
// Bad 🗨️
function Component(props) {
  const aProp = { someDynamicProp: 'someValue' }
  return <AnotherComponent style={{ margin: 0 }} aProp={aProp} />
}
```

```
// Good 😊
const styles = { margin: 0 };
function Component(props) {
  const aProp = { someDynamicProp : 'someValue' }
  return <AnotherComponent style={styles} {...aProp} />
}
```

Anonymous function

```
// Bad 🗨️
function Component(props) {
  return <AnotherComponent onChange={() => props.callback(props.id)} />
}

// Good 😊
function Component(props) {
  const handleChange = useCallback(() => props.callback(props.id), [props.id]);
  return <AnotherComponent onChange={handleChange} />
}

// Good 😊
class Component extends React.Component {
  handleChange = () => {
    this.props.callback(this.props.id)
  }
  render() {
    return <AnotherComponent onChange={this.handleChange} />
  }
}
```


Prevent rebuild component

```
// Bad 🗨️
// 避免對大型的組件頻繁的建立與刪除
function Component(props) {
  const [view, setView] = useState(true);
  return view ? <SomeComponent /> : <AnotherComponent />
}

// Good 😊
// 使用該方式提升性能和速度
const visibleStyles = { opacity: 1 };
const hiddenStyles = { opacity: 0 };
function Component(props) {
  const [view, setView] = useState(true);
  return (
    <React.Fragment>
      <SomeComponent style={view ? visibleStyles : hiddenStyles}>
      <AnotherComponent style={view ? visibleStyles : hiddenStyles}>
    </React.Fragment>
  )
}
```

React.Fragment

```
// Bad 🗨️
function Component() {
  return (
    <div>
      <h1>Hello world!</h1>
      <h1>Hello there again!</h1>
    </div>
  )
}

// Good 😊
function Component() {
  return (
    <React.Fragment>
      <h1>Hello world!</h1>
      <h1>Hello there again!</h1>
    </React.Fragment>
  )
}

// Very Good 😊
function Component() {
  return (
    <>
      <h1>Hello world!</h1>
      <h1>Hello there again!</h1>
    </>
  )
}
```


學習不需要為公司、長官或同事，不需要為別人，只為你自己。

五倍紅寶石 高見龍

奶綠茶的粉絲團