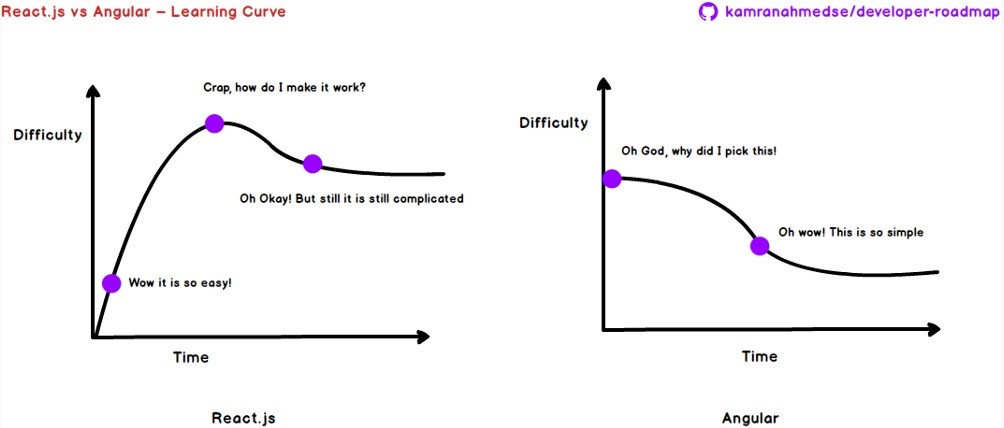
**什麼是Angular?**

Angular是一個前端開發框架。

它能幫你更輕鬆的構建 Web應用程式。 Angular 集聲明式模板（declarative templates）、依賴注入（dependency injection）、端到端工具和一些最佳實踐方式於一身，為您解決開發層面的各種挑戰。 替開發者提升構建 Web、手機或桌面應用程式的能力。

Angular 有提供非常多功能給開發者使用，如：

* i18n - 多國語系
* Animation - 動畫
* Router - 路由
* CLI - 命令列工具
* Material - Google 力推的 Material 網頁設計框架mat
* Augury - 官方出版的 Google Chrome 開發者工具
  1. Angular與React 學習曲線



上圖為Angular與React的學習曲線，Angular 一開始的學習曲線雖然與其他框架相比高了些，畢竟它比較囉嗦，開發方式也跟傳統的開發方式大相逕庭。

對於一般習慣傳統開發方式或是剛踏入前端甚至是程式領域的人來說，是滿難以理解的。

但是正如前面所講到，Angular 跟其他的開發框架不一樣的是，它比較像是一個完整的平台，從開發到測試它都一手包辦，並且擁有豐富的生態系，

背後有 Google 在推動，甚至就許多公司的立場而言，Angular 有完整、完善的規範與一致的開方式，這才是一個框架、一個平台真正的價值。

**環境建置**

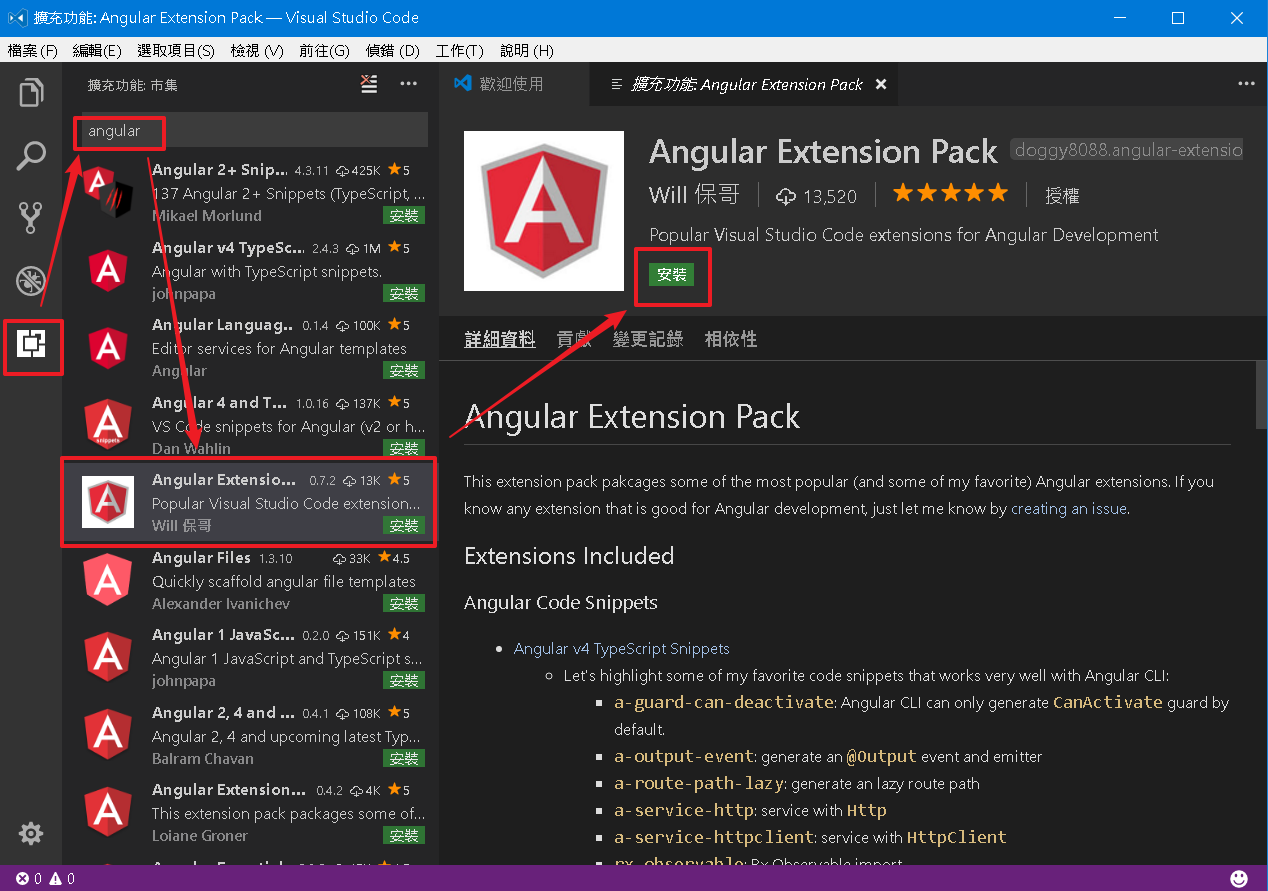
使用Angular開發前端需要這些工具：

* Visual Studio Code

建議至官方網站下載：[https://code.visualstudio.com/。  
\\pf03\SOURCE\國泰世華專用\VSCodeSetup-x64](https://code.visualstudio.com/。\\pf03\SOURCE\國泰世華專用\VSCodeSetup-x64%20) 也有安裝檔案，但有可能版本較舊。

下載及安裝完成後，建議安裝擴充套件如下圖。

(2-1) vs code Angular擴充套件



Visual Studio Code可設定佈景主題(左下角齒輪)

(2-2) 設定佈景主題



(2-3)



(2-4 Visual Studio Code 快捷鍵)

|  |  |
| --- | --- |
| 全部儲存 | ctrl + k, s |
| 切換關聯檔案 | alt + o |
| 快速開啟檔案 | ctrl + p |
| 開啟/關閉左邊的 siderbar | ctrl + b |
| 開啟/關閉檔案總管 | ctrl + shift + e |
| 開啟/關閉搜尋視窗 | ctrl + shift + f |
| 開啟/關閉原始檔控制(git) | ctrl + shift + g |
| 自動排版 | alt + shift + f (記得設定 prettier.singleQuote) |
| 註解 | ctrl + k + c |
| 單行註解 | ctrl + / |
| 取消註解 | ctrl + k + u (or ctrl + /) |

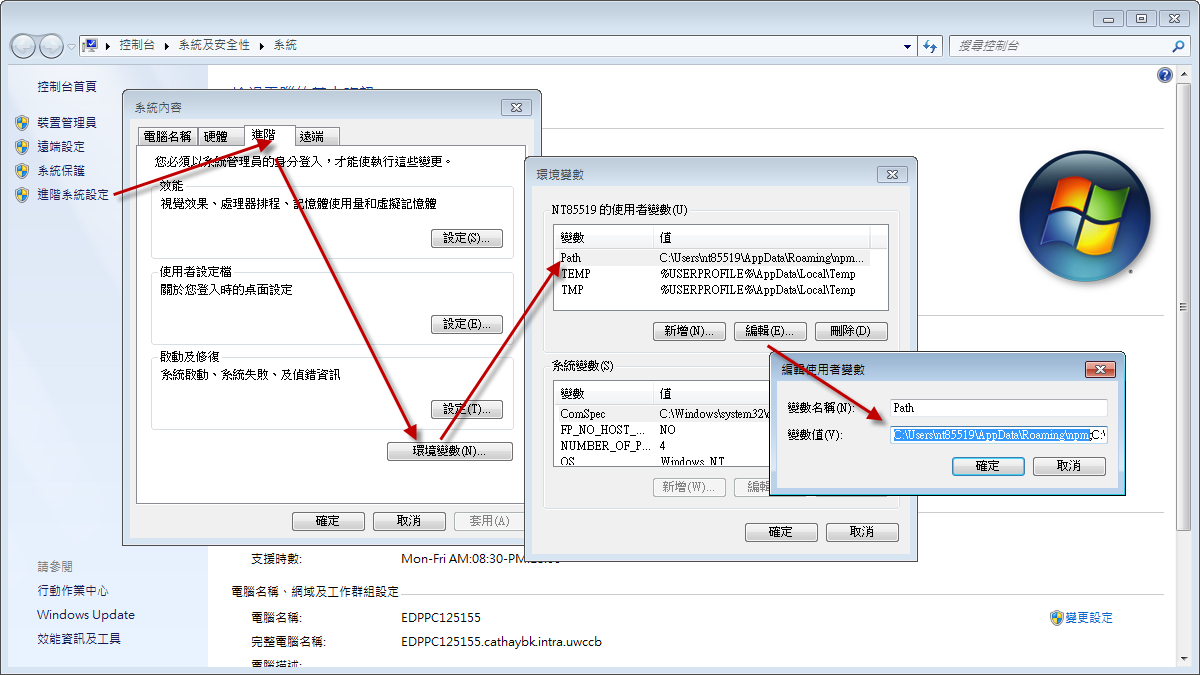
更多Visual Studio Code快捷鍵可參考附件： 

* Node.JS

下載路徑：[\\pf03\SOURCE\國泰世華專用\node.js](file:///\\pf03\SOURCE\國泰世華專用\node.js)

此安裝會一併安裝NPM(Node Package Manager)，說明請參考npm分頁  
安裝完成後，**必須設定套件環境變數**

（3-1）設定環境變數

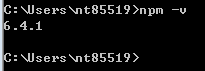


安裝完成後，使用CMD確認是否安裝成功及版本資訊。

(3-2)



(3-3)

****

**行內安裝NPM套件：**

因行內環境無法與外部環境連線，使用npm安裝套件時須使用行內nexus repository

使用方法有兩種:

* 於安裝時指定使用nexux repository

> npm install -g @angular/cli --reg http://88.8.111.206:8081/nexus/content/groups/npm-all/

* 設定npm 之後安裝皆會使用nexux repository

> npm config set registry http://88.8.111.206:8081/nexus/content/groups/npm-all/

刪除設定

> npm config delete registry

NPM 指令相關

npm安裝指令 npm install 中的參數 -g 為設定為全域(global)，沒設定全域將會安裝在目前資料夾

完整安裝 指令為 npm install –global 可簡化為 npm i –g

**NPM – PROXY**

**此方法為非正常管道，非必要不建議使用。**一開始可先不使用

設定PROXY

> npm config set https-proxy http://pxwsg:80

> npm config set proxy http://pxwsg:80

> npm config set strict-ssl false

解除設定

> npm config delete https-proxy

> npm config delete proxy

> npm config delete strict-ssl

建議一開始便可進行軟體安裝

CyberArk(端點防禦系統/APT)

如有覺得NodeJS/NPM使用上有太慢的情況，請更新APT：

[\\pf03\SOURCE\國泰世華專用\CyberArk端點防禦系統\APT-PCAgent\Win10-Taiwan\x64](file:///\\pf03\SOURCE\國泰世華專用\CyberArk端點防禦系統\APT-PCAgent\Win10-Taiwan\x64)

* Angular CLI

Angular cli 則為 Angular 提供的工具，開發人員可使用此工具 CLI => Command Line快速建立Angular開發專案

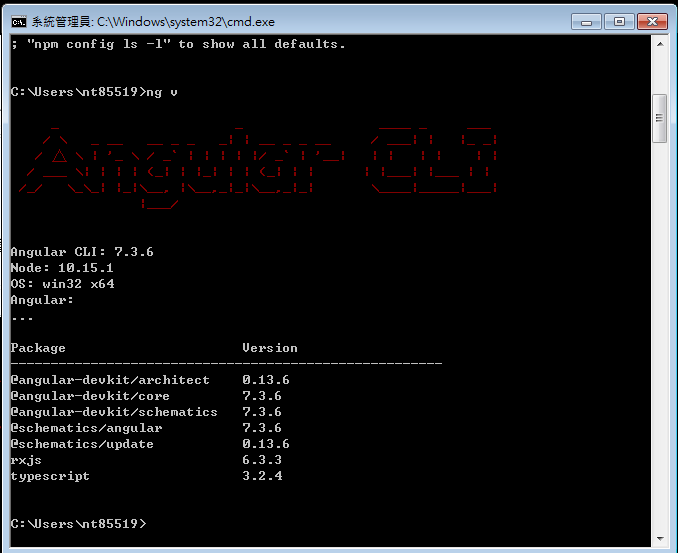
於CMD輸入npm install -g @angular/cli 安裝Angular CLI

(4-1)



使用ng v查看Angular CLI是否成功安裝及安裝版本。

(4-2)



Angular CLI各版本指令參數及操作方式略有不同，最新版可參考[官方文件](https://cli.angular.io/)

Angular CLI相關語法說明參考：[Angular深入淺出三十天](https://ithelp.ithome.com.tw/articles/1020548)

更新Angular CLI置最近版本:

CMD輸入> npm update -g @angular/cli@lastest

如無法直接update 請先移除後再重新安裝

> npm uninstall -g @angular/cli

> npm i -g @angular/cli@latest

**相關可參考連結：**

[iT邦幫忙 - [Angular 深入淺出三十天]](https://ithelp.ithome.com.tw/articles/10203185)

[Jonny Huang 的學習筆記](file:///C:\Users\nt85519\Desktop\教學文件\Jonny%20Huang%20的學習筆記)

[開發二科 – Angular技術交流](http://pispap01/study/angular/_layouts/15/start.aspx#/SitePages/%E7%A4%BE%E7%BE%A4%E9%A6%96%E9%A0%81.aspx)

[Angular官方](https://angular.io/)

[Angular – 中文](https://angular.tw/tutorial)

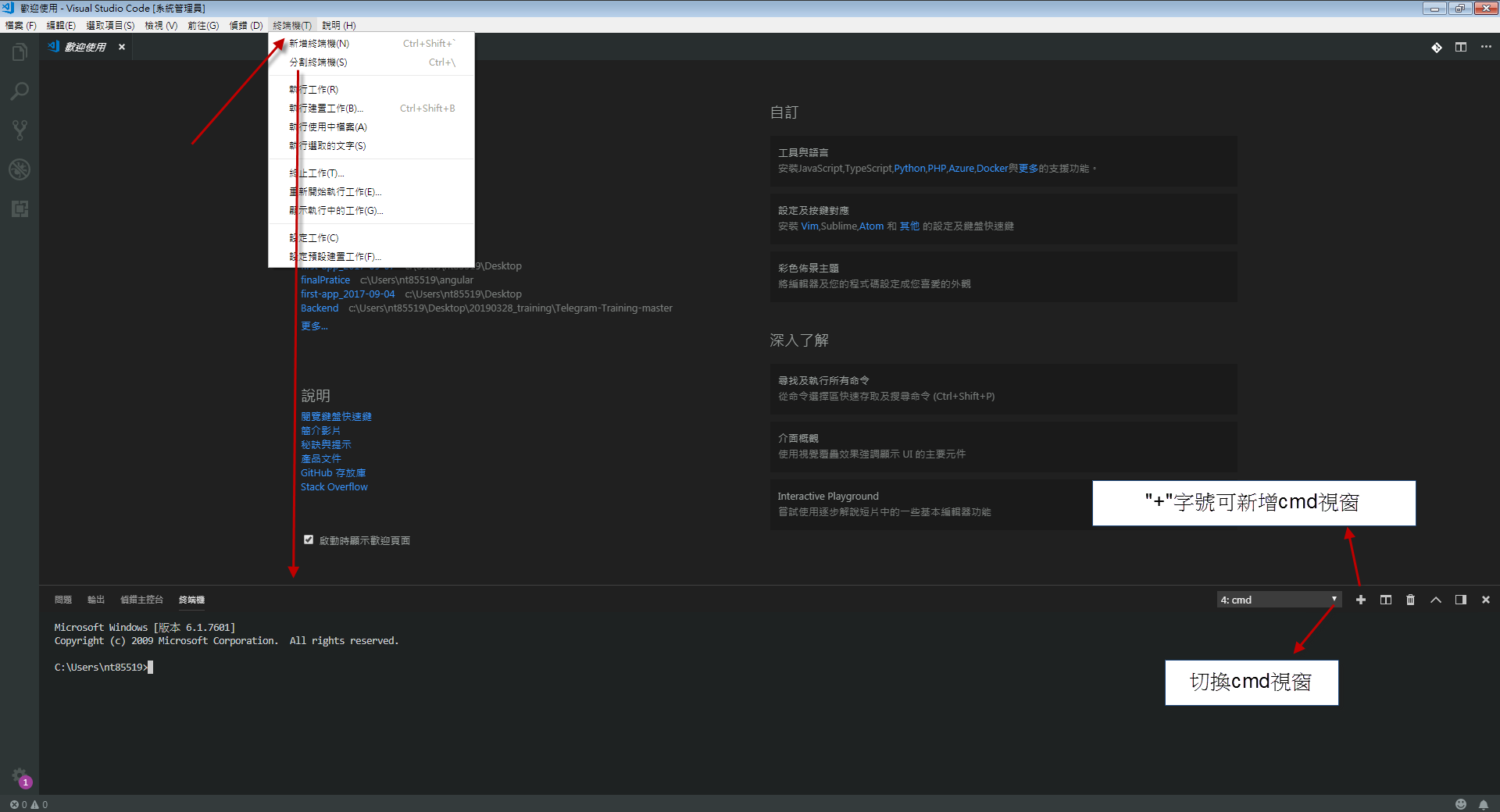
[Angular CLI generate 簡寫](https://github.com/angular/angular-cli/blob/master/docs/documentation/generate/component.md)

**建立專案**

**\***可以透過Visual Studio Code使用Angular CLI建立新專案：

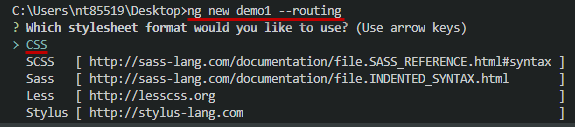
開啟VS Code，快捷鍵 Ctrl + ` 或是終端機、新增終端機 開啟CMD，切換至正確位置，

(5-1)



> ng new [ProjectName] --routing

(5-2)

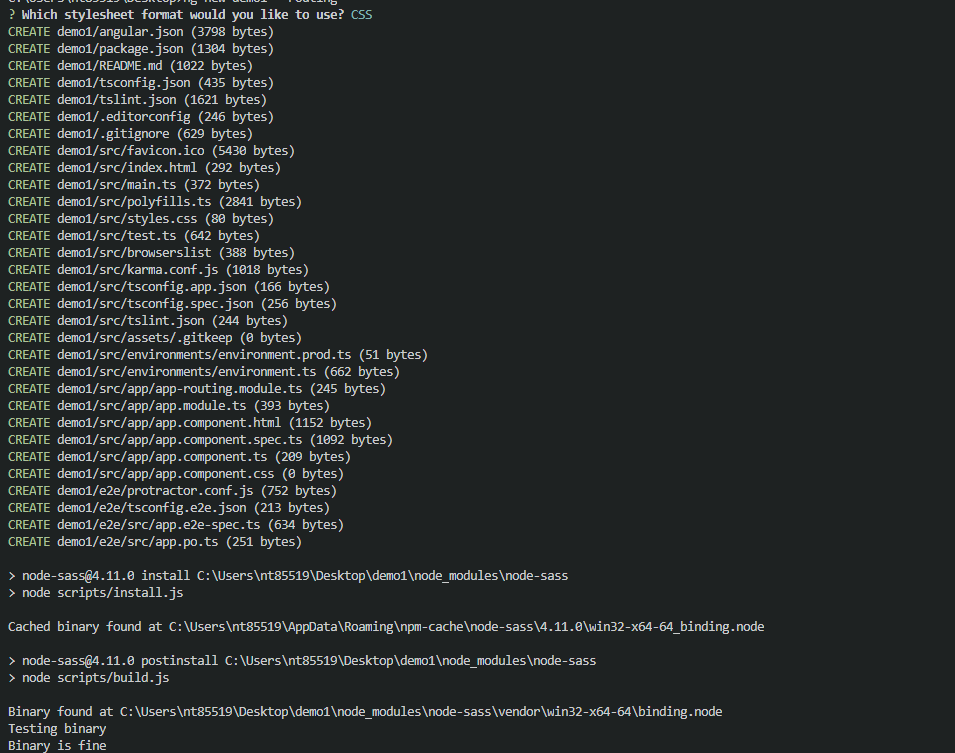


建立專案時會詢問預設css樣式，此處使用CSS樣式

--routing：參數表示同步建立路由模組，後續再另行說明。  
--style scss：預設樣式是採用css格式，此處改為scss。

Angular會自動建立需要檔案，

(5-3)

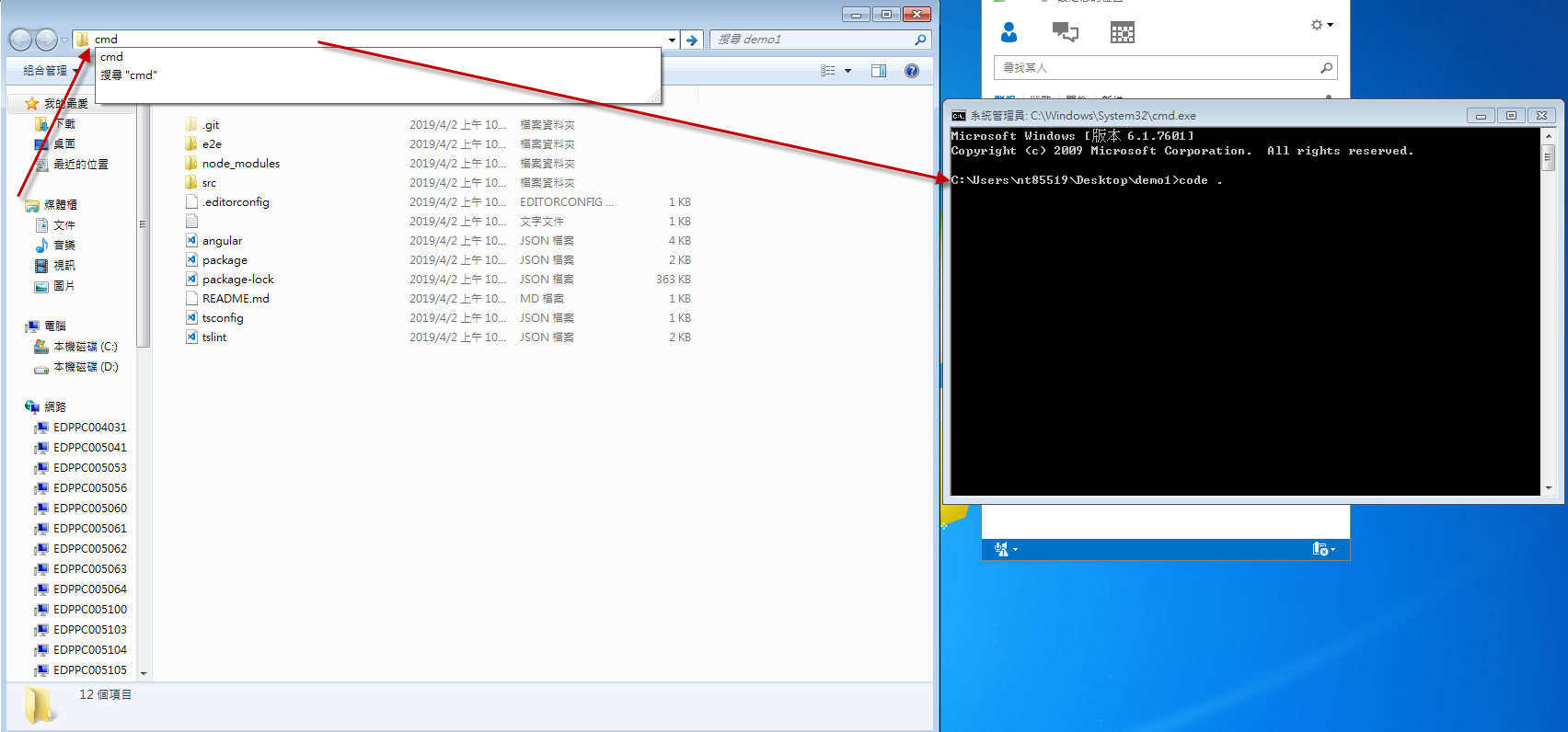


VS Code開啟專案方式：

1. 開啟專案資料夾 -> 在路徑輸入 “CMD” Enter -> 在CMD輸入 “code .” -> VS code即可自動開啟

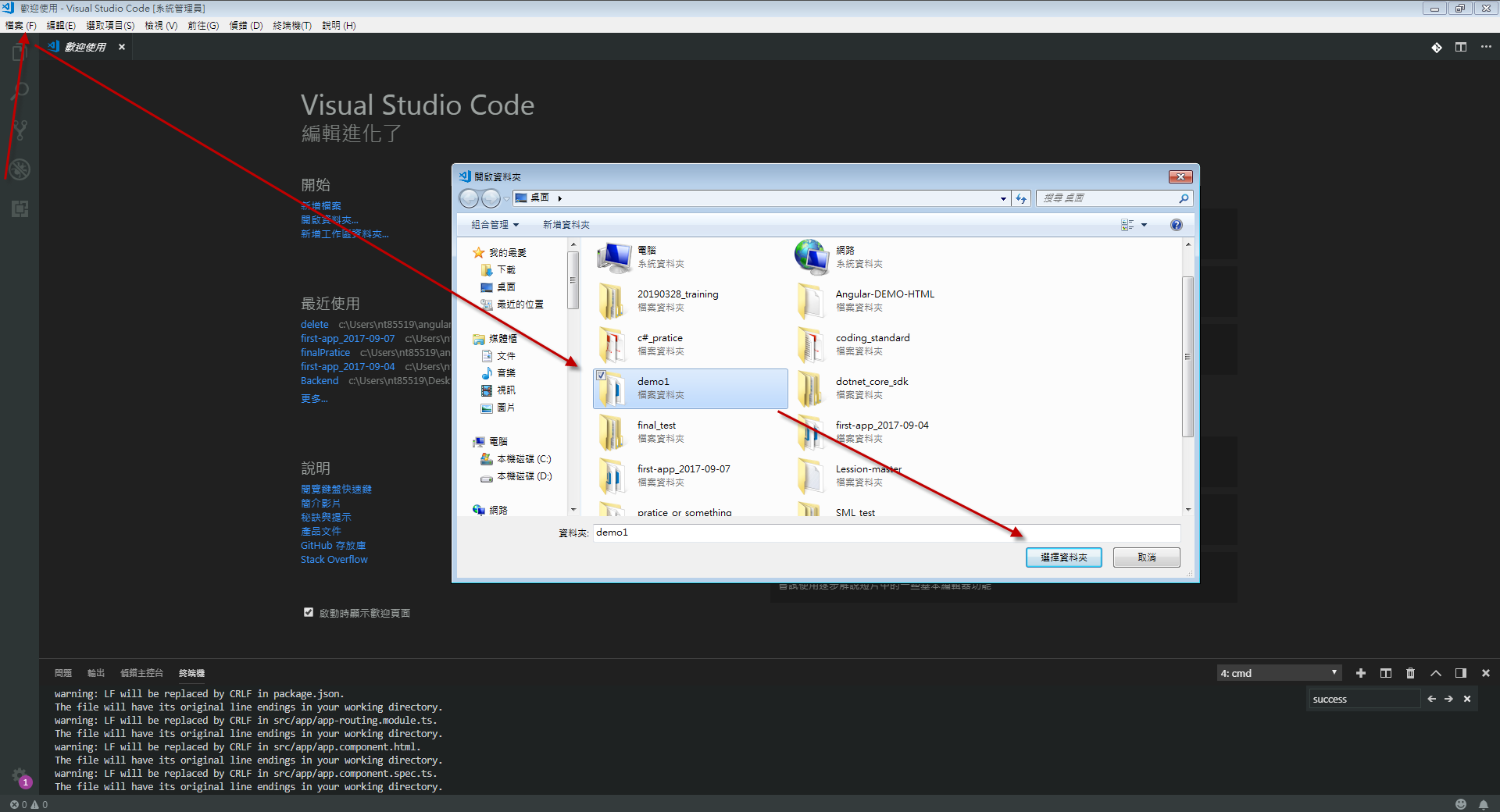
註\* 如果 CMD不認識 code指令，請確認VS Code環境變數是否有正確加入

(5-4)



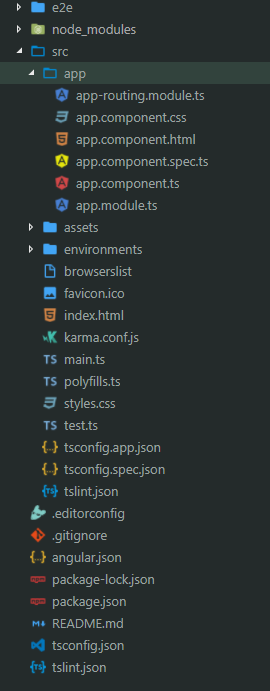
2. 直接使用VS Code開啟專案資料夾

(5-5)



透過 VS Code 可以發現 CLI 自動幫我們建立好了開發過程所需要用到的檔案。

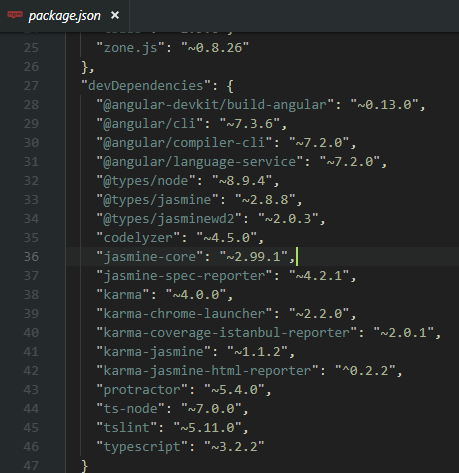
(5-6)



**Package.json**

表示執行環境所需的package

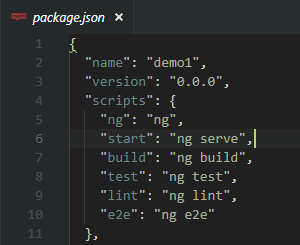
(5-7)



**Package.json**

scripts 表示客製化的參數與對應的指令，可以透過 npm run [parameter] 執行，例如：npm run start

(5-8)



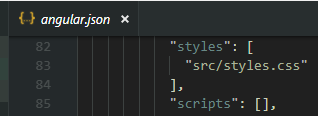
package 可能會隨功能強化或是bug修正而發布新版本，尤其是 Angular 有這大量的開發人員，因此更新速度特別快，但是新版本與舊有版本是否相容就有待查證，因此若要使用新版本，建議在剛建立完專案時就先升級，既有專案若要升級，建議做好版本控制或是備份。

升級示範：  
透過**命令提示字元**使用 npm i [package name] 重新安裝 package。

**Angular.json** 為 Angular-CLI 的設定檔

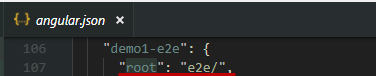
Styles、Script為專案預設引入的CSS檔案及CSS檔案

(5-9)



**e2e/**為專案程式碼目錄，**不建議修改**

(5-10)



**Development server**

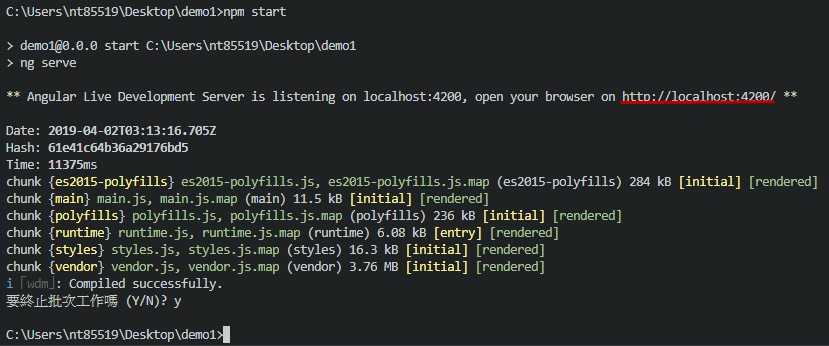
Npm start 或是 ng serve都可以啟動專案 (請參考5-8)

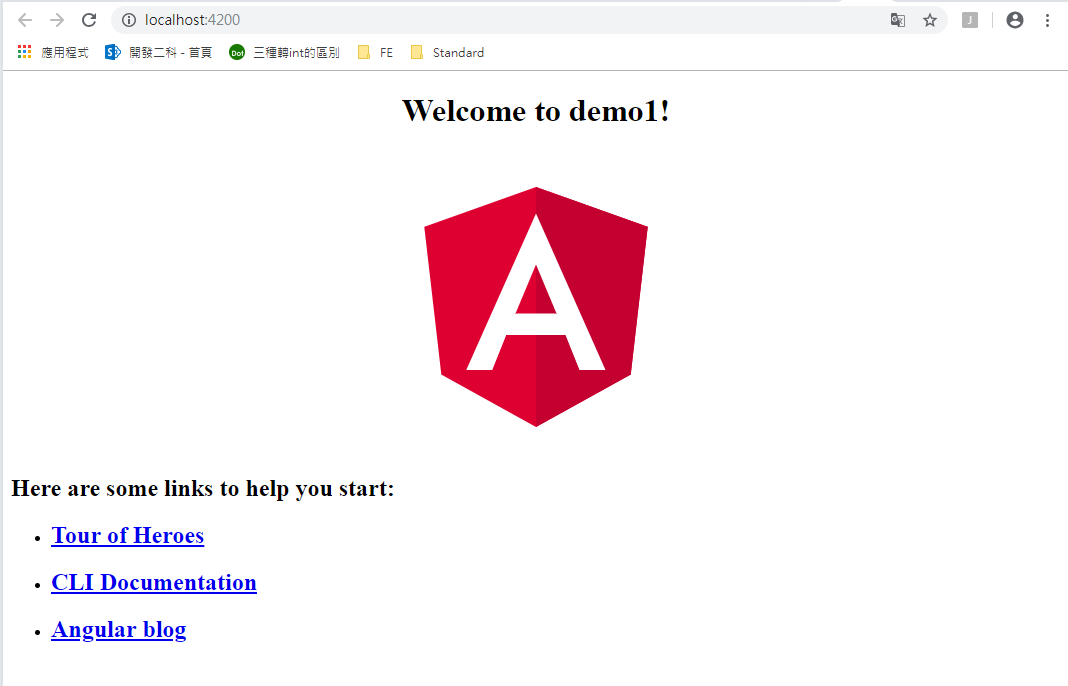
(當專案已經成功start，修改程式碼時不必restart，會自動更新在瀏覽器上)

透過 ng serve 指令可以執行開發用的測試服務器，並可透過 http://localhost:4200/ 網址來測試

可透過 Ctrl + c 來終止服務。

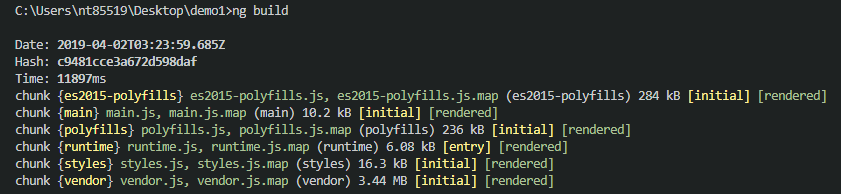
(5-11)

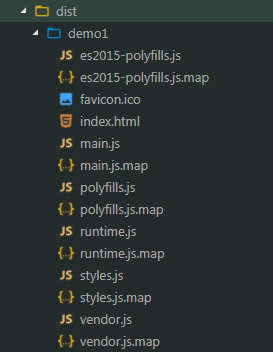




#### Build

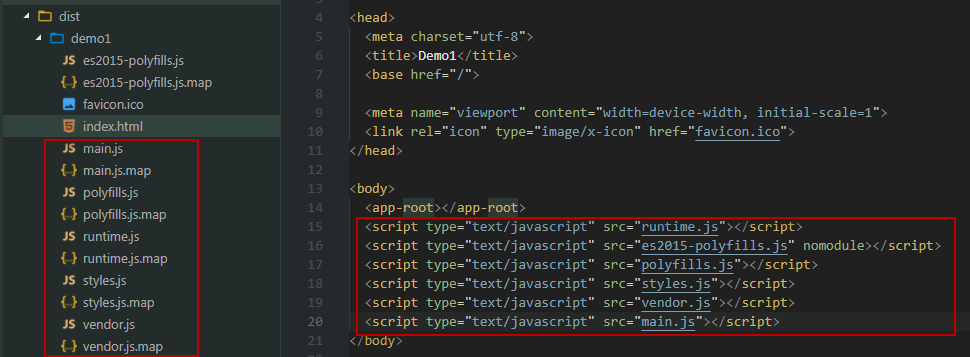
透過 ng build 指令 CLI 會在專案目錄下建立 dist 資料夾，並將編譯後的網站放置於此，若要將專案佈署至正式環境時，記得改用 ng build -prod 編譯。



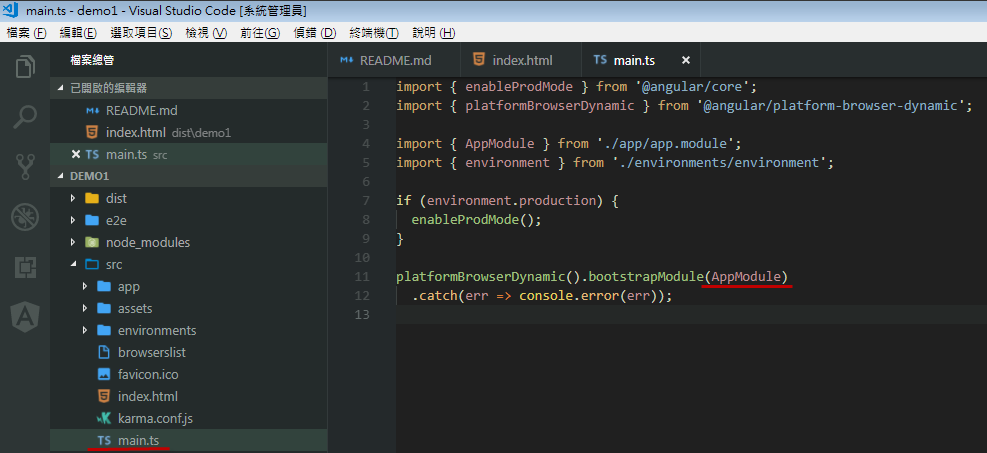


## Angular 基本運作流程

正常運作下，我們對網站發出請求後網站會回傳 index.html，而在 index.html 內會包含 main.ts 的 JavaScript。

由之前 ng build 編譯所產生的 index.html 可以看到網頁載入了 main.bundle.js 網頁載入了。  


開啟專案目錄下 src\main.ts 檔可以發現 Angular 將 AppModule 當作起始模組載入。

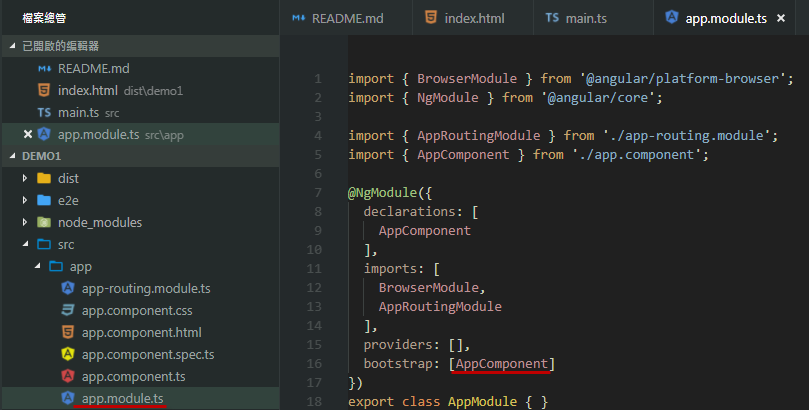


**流程：**index.html > main.ts > app.module.ts

Angular 是模組(NgModule)為基礎的高度元件化框架，可在模組內可以再包含模組

我們可以依功能將模組再分拆成數個小模組，方便維護，Angular 在執行時只會有一個起始模組。

由 **AppModule**(src\app\app.module.ts) 可看到在 @NgModule 內設定 bootstrap 屬性為 **AppComponent**，也就表示將 **AppComponent** 指定為**起始元件**。



所以流程又變成了：

**流程：**index.html > main.ts > app.module.ts > app.component.ts

Angular Component

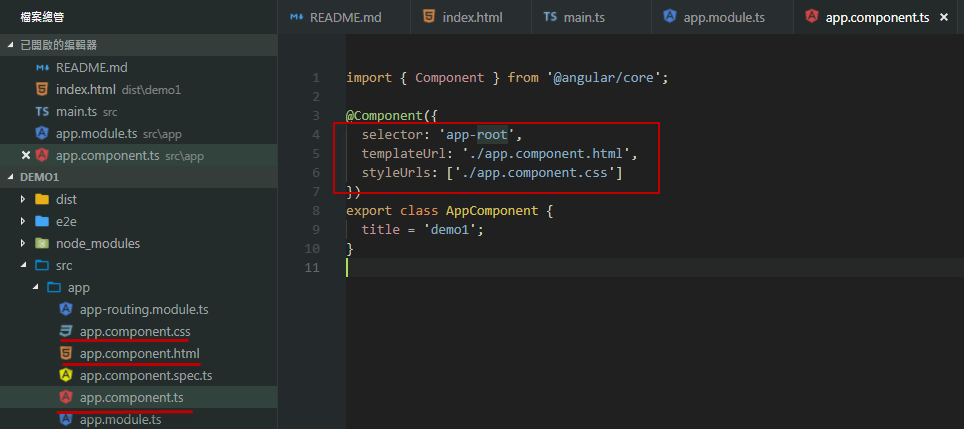
Angular 的介面呈現主要是透過元件(Component)來處理，而元件需依附於模組(NgModule)，因此正常會在**起始模組**去指定**起始元件**。  
@expression 是 TypeScript 的裝飾器，此種聲明可以將其附加到宣告的對象內，Angular 常用的裝飾器有 @NgModule、@Component、@Injectable、@Directive。

開啟專案目錄下 src\app\app.component.ts 檔可以發現 **AppComponent** 這個類別宣告了一個 @Component 的裝飾器，所以 **AppComponent** 被視為一個 Angular Component。

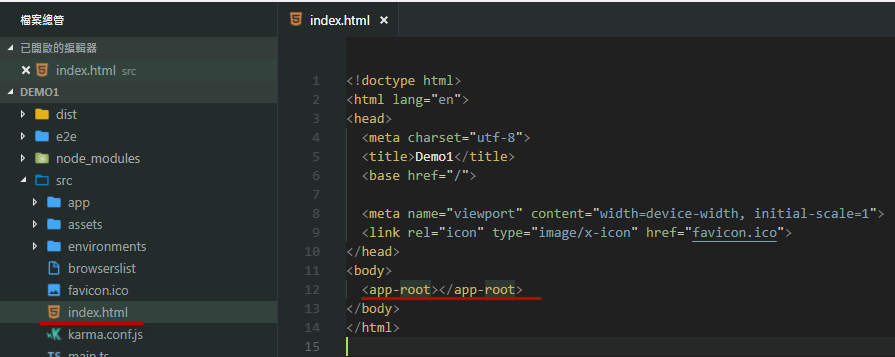
屬性templateUrl值為 './app.component.html'，表示此元件會以 ./app.component.html 當作樣板輸出至介面上

屬性styleUrls 屬性值為 ['./app.component.css']，表示元件輸入的樣板會套用 ./app.component.css 樣式

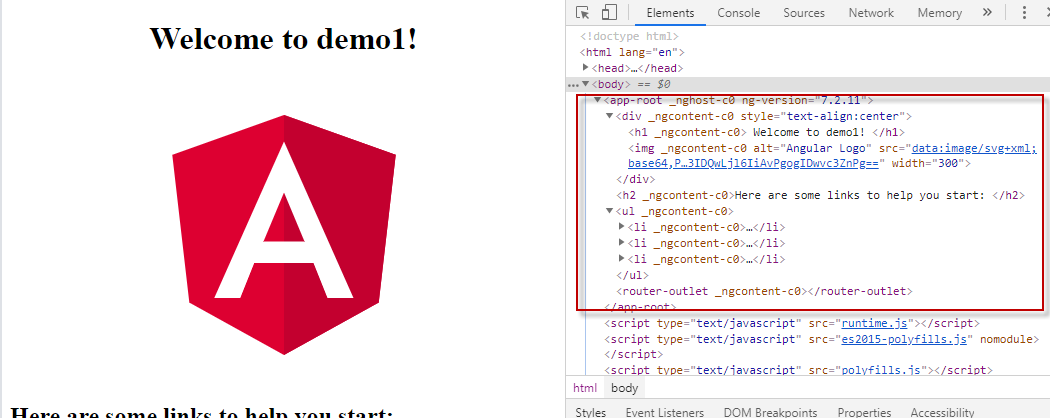
屬性 selector 被指定為 'app-root' ，此屬性類似 CSS 的選擇器，會將網頁內的有包含 app-root 的 tag 都替換成 ./app.component.html。



開啟專案目錄下src\index.html 檔可以發現內含一個 app-root tag，所以正常來說 Angular 在運作時就會被替換成 app.component.html。



從瀏覽器開發工具檢視，app-root tag 被插入了app.component.html。

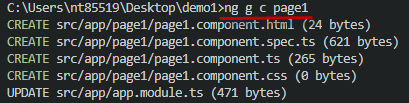


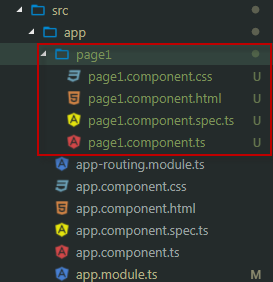
## 如何新增元件

要建立新的元件可以**手動建立相關檔案並修改相關設定**，但是另一個更好的方法就是透過 CLI 幫我們產生相關檔案，語法如下：  
ng generate component [name] => generate可以縮寫成g，component可以縮寫成c =>

> ng g c page1

透過指令 ng g c page1 可以發現 CLI 幫我們建立一個與元件同名(page1)的資料夾，並在資料夾內產生4個檔案，最後再更新 AppModule。





Page1.component.css => Page1Component所對應的CSS

Page1.component.html=> Page1Component所對應的樣板

Page1.component.spec.ts => Page1Component所對應的單元測試檔

Page1.component.ts => Page1Component所對應的 已宣告 @Component 裝飾器的類別，此處也是主要程式撰寫的地方。

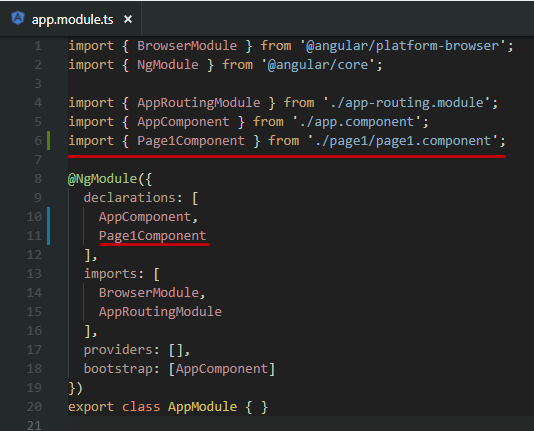
若不想要建立資料夾，可以加上 --flat 參數。

若是不想建立 style 檔，可以加上 -is 參數。

若是不想建立 template 檔，可以加上 -it 參數。

相關參數請參考 [此處](https://github.com/angular/angular-cli/blob/master/docs/documentation/generate/component.md)。

打開 app.module.ts 可以看到CLI自動將Page1Component註冊到AppModule中

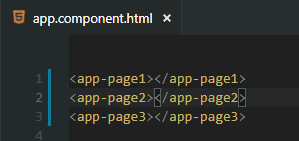


Component 無法獨立存在，必須依附在 NgModule 內，依附方式就是將 Component 註冊到 NgModule 的 **declarations** 屬性內。

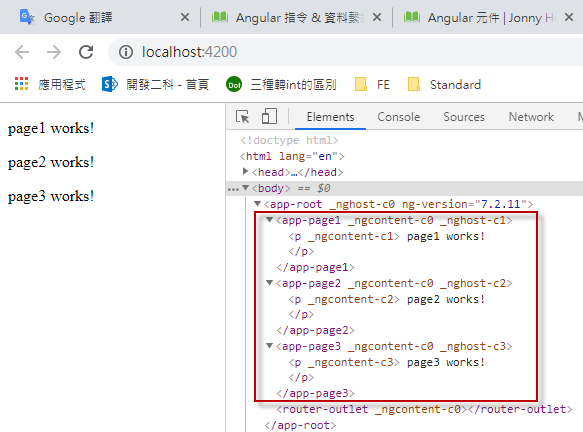
請試著練習產生Page2與Page3。

### 套用 Component

開啟 app.component.html，並將 Page1Component、Page2Component、Page3Component 內 selector 所設定的 tag 加入至 app.component.html。請注意先別將最後一行<router-outlet></router-outlet>刪除，在之後將會說明此用法。



使用npm start透過瀏覽器查看，並發現剛剛所新增的三個Component已經顯示在畫面中



## Angular 路由器

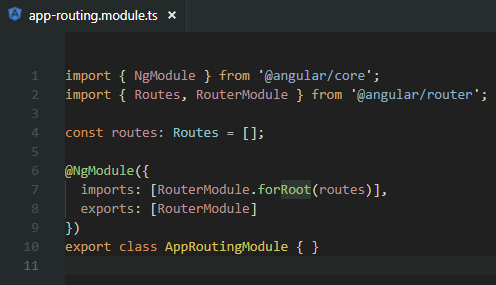
## 路由 (Routing)

在 Angular 元件 最後面我們練習了直接透過 tag 來嵌入 Component，但是這只是在頁面一開始呈現的效果，而正常情況下還需要依照使用者的操作適當的改變內容，例如切換不同功能時需要整個頁面替換掉，做查詢時可能要在介面上的某個區塊將結果顯示出來。

Angular 提供了一個路由機制，讓我們可以很方便的抽換部分區塊，說是部分抽換區塊是因為正常情況下，Angular 的根元件(起始模組內所指定的起始元件)在一開始載入後便無法被抽換掉，我們只能替換根元件內的內容。

在 建立 Angular 專案 我們在建立專案時透過指令 ng new [NAME] --routing建立專案，其中帶了一個參數 --routing，此參數讓 CLI 在建立專案同時幫我們新增一個具有路由功能的模組― AppRoutingModule，並將該模組註冊到 AppModule 內。

開啟 scr\app\app-routing.module.ts 可以發現裡面宣告一個 routes 的陣列變數讓我們可以加入多個路由規則，最後 RouterModule.forRoot(routes) 則是將此路由變數宣告為根路由。

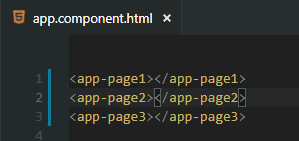


## 路由插座 (router-outlet)

剛剛專案內的 app.component.html，網頁內的最後一行是一個奇怪的 tag- <router-outlet></router-outlet>，我們可以把 router-outlet 看成 Component 的 tag，因為 Angular 的路由機制運作結果會如同 Component 的 tag，會以這個 tag 當作標記去插入指定的 Component。

## 練習編寫路由規則

移除剛剛新增的<app-page1></app-page1>~<app-page3></app-page3>留下 <router-outlet></router-outlet>。

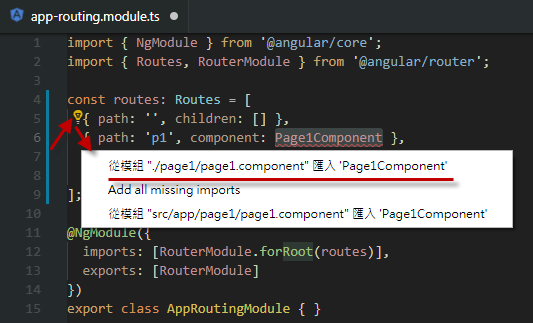


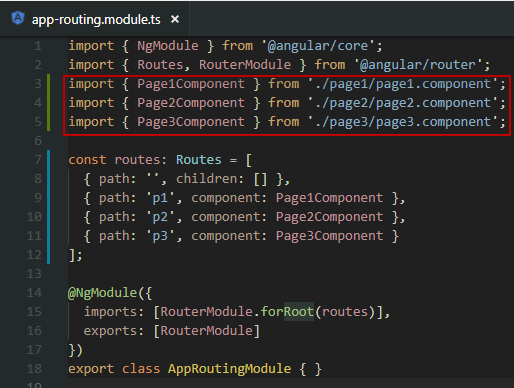
開啟 app-routing.module.ts ，加入3個路由規則分別對應到 Page1Component、Page2Component、Page3Component。

* path：設定導覽的相對路徑
* component：設定要載入的 Component

當導覽路徑變化時 Angular Router 會將導覽路徑拿來跟路由規則逐一比對，如果路徑與 path 條件符合就會執行該路由條件，我們目前設定了 component 屬性，所以符合條件時就會將 component 屬性所指定的 Component 插入目前 Component 的 router-outlet tag 內。

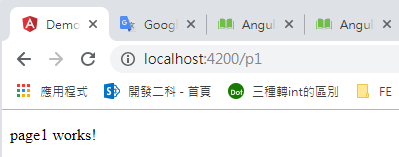
在模組中，如果要使用其他元件，都必須要有import的動作，如果在使用元件的時候出現紅字，可以把該元件反白起來，行列左邊會出現小鈴鐺，可以自動做import (前提是必須真的有該Component)。

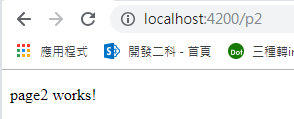


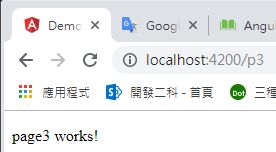


查看瀏覽器，分別連結至 <http://localhost:4200/p1>、<http://localhost:4200/>p2、<http://localhost:4200/>p3

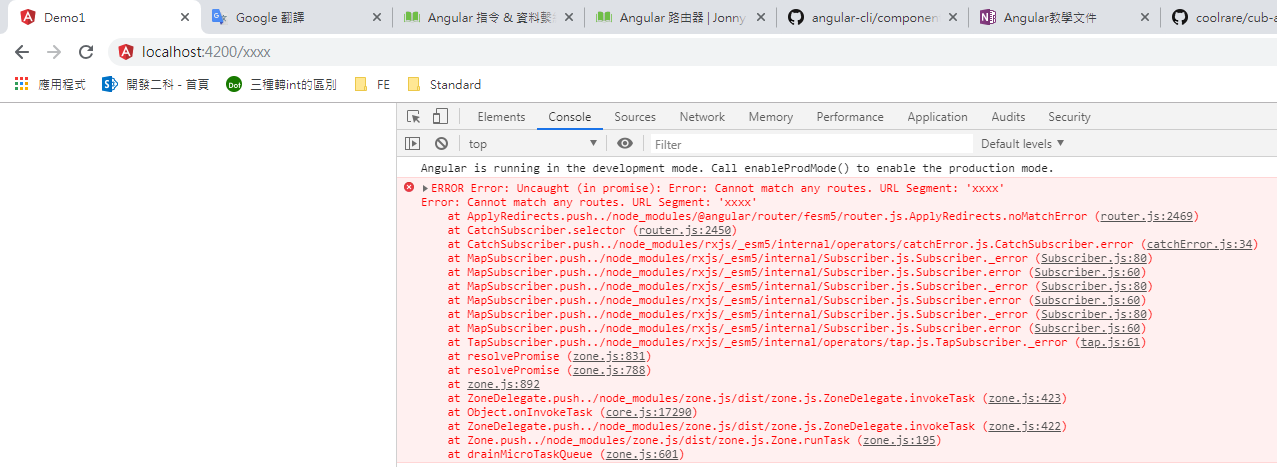
會看到各網址已經對應到各Component內容。







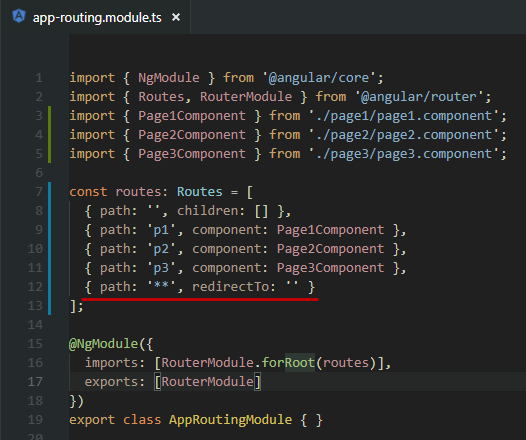
嘗試輸入不存在的路由規則，例如：http://localhost:4200/xxxx，可以看到 Angular 好像自動幫我們導回到http://localhost:4200，可是開啟瀏覽器開發者工具發現系統發出錯誤的訊息



## 萬用路由 (\*\*)

Router提供萬用路由規則，如果path設定為 ‘\*\*’ 表示條件為任意值，當路由比對此規則時，不管url是什麼，必定符合。所以在萬用路由後的規則基本上是無效的，所以正常會將萬用路由放置在路由規則的最後面。

加入一個萬用路由，並改用 redirectTo 屬性，將其設定為‘p1’。



此時路徑後方不管輸入什麼都，只要不是符合前面的規則，例如：<http://localhost:4200/p4、http://localhost:4200/xxxx、http://localhost:4200/everywhere>，

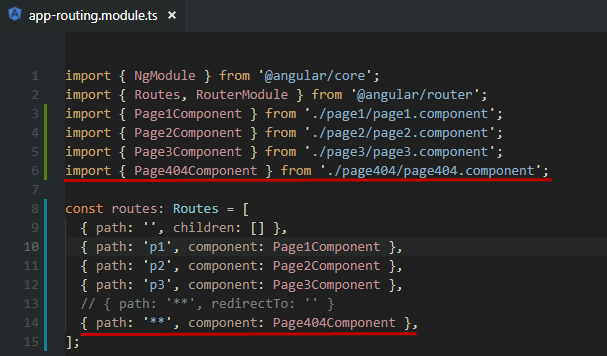
都會被重新導引到首頁(http://localhost:4200/)，從瀏覽器開發者工具並不會看到任何錯誤訊息。

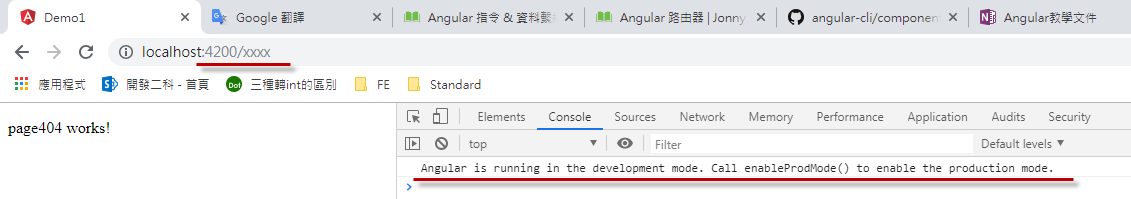
一般而言萬用路由不應該被執行到，這表示目前導覽路徑是一個非預期的路徑，所以萬用路由會比較像是為了避免系統出錯的預防機制，比較常見的情境是

* 導覽至首頁：此種情境適合在發生錯誤時使用者可以捨棄目前資訊，大部分來說這種非預期狀況是一般使用者無法排除的，因此導引至首頁可以讓使用者繼續操作，在 PWA(Progressive Web App) 網站應該是不錯的選擇。
* Page Not Found：目前網站比較普遍的作法是建立一個 HTTP 404 頁面，對於比較需要立即修正的網站會比較適合，除了可以藉由 404 頁面蒐集當時狀態讓後續可以分析了解原因，這種中斷式頁面也較容易讓使用者立即反應回饋給維護人員。

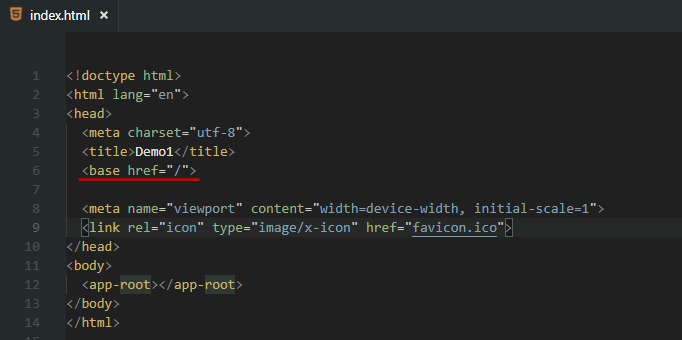
### 練習製作 HTTP 404 頁面

透過指令 ng g c page404 建立一個，並修改 src\app\page404\page404.component.html 內容如下：





可以看到網頁正常運作，Page404內容成功呈現，並且保留原輸入網址。

我們可以發現路由規則其實都是在比較相對路徑，我們在 path 並不會輸入完整路徑，Angular 又要如何知道應用程式的根路徑？  
開啟 index.html 檔會發現 head 區間內包含了一個 <base href="/">，Angular Router 便是透過 base 這個 tag 來依序組合導覽路徑。

# 導航 (Navigation)

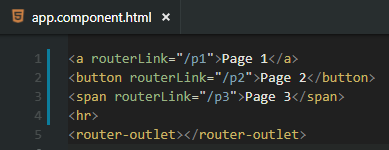
在剛剛我們透過改變url路徑切換內容，但是一般使用者不會透過修改網址切換頁面，都是透過點選操作。

RouterLink

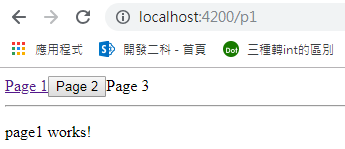
可以直接在樣板內直接設定連結對象。

練習加入 RouterLink

開啟app.component.html加入以下html，並注意其中的routerLink



開啟 <http://localhost:4200> 並點選三個元件，都會導覽至各頁面中。



# 模組 (NgModule)

過去我們將所有元件(Component) 都import到 **AppModule** 內，但是實務上的專案可能會有幾十個甚至上百個元件，如果全部都放置在 **AppModule** 內，會造成專案難以維護也難以分工的問題，所以比較合理的做法就是將元件拆分至不同的 模組(NgModule) 內，再將模組聚合起來。  
開啟 src\app\app.module.ts 可以看到目前所建立的元件。



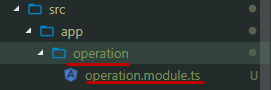
我們可以把Module假想成資料夾，把Component想成是檔案。

## 建立新模組

> ng g m operation (m為module簡寫)

產生了一個operation的資料夾與一個module.ts檔



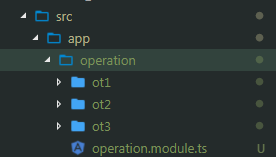


在此operation底下建立三個Component

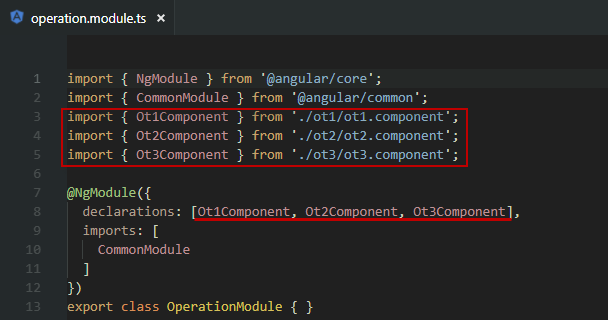
> ng g c operation\opt1

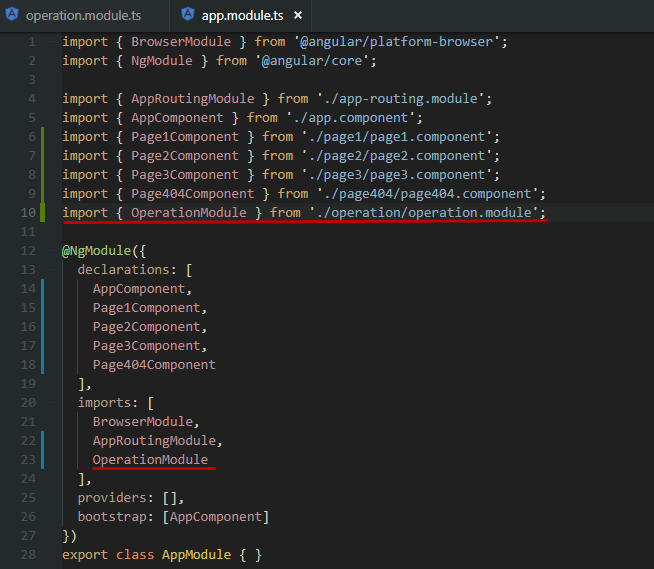
> ng g c operation\opt2

> ng g c operation\opt3



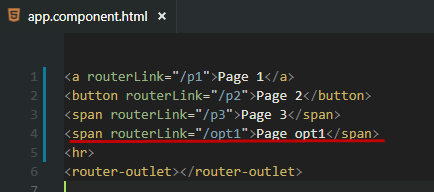
點開 operation.module.ts 可以發現opt1、opt2、opt3都已自動註冊到operation中，由此可知CLI在建立Component會將此元件註冊到最近的module中。



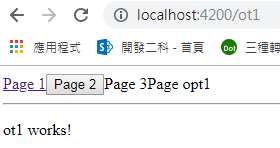
將OperationModule註冊到AppModule(根模組) 

在將OperationModule裡中需要使用的Component加到app.component.html、app.component.ts

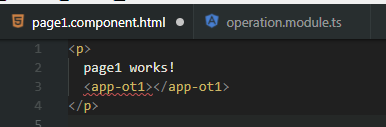




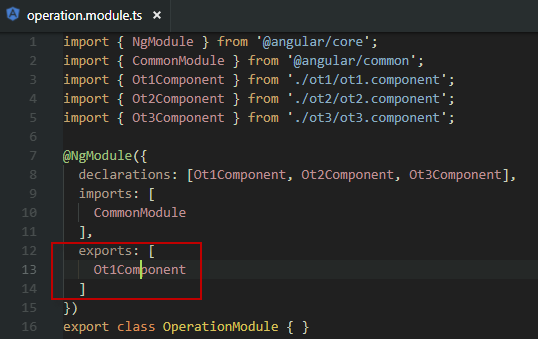
開啟瀏覽器，ot1 Component已經運作



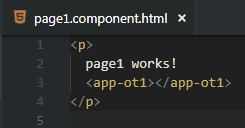
如果Page1Component 需要使用到 Opt1Component，所以正常方式是將 Opt1Component 的 tag (app-opt1) 加到src\app\page1\page1.component.html 內，執行看看結果，這邊會發現 Angular 不認得 app-opt1 這個 tag。



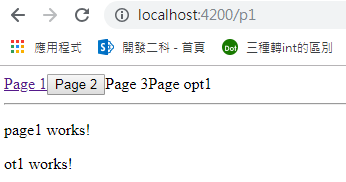
雖然已經將opt1的模組 OperationModule註冊到AppModule中，但是**如果需要直接使用，必須在自己的Module中exports(匯出)這個元件出來使用。**



Exports後，AppModule – Page1.component.html，已經認識app-ot1這個tag



開啟瀏覽器檢視，ot1也成功顯示於p1。



NgModule 的 imports 提供了匯入功能，讓我們將要使用的元件匯入到此模組內，而 exports 則提供匯出功能，讓別的模組可以知道此模組提供什麼元件，所以正常要 exports 的元件也一定要先匯入進來，因為是元件所以匯入是加到 declarations 屬性內。

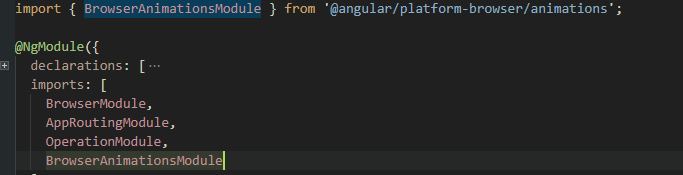
## [Angular UI：Angular Material](https://material.angular.io/)

[其他Angular開發好幫手](https://angular.io/resources)(僅供參考)

[Material 官方安裝步驟](https://material.angular.io/guide/getting-started)

> npm install --save @angular/material @angular/cdk @angular/animations

將 動畫模組(BrowserAnimationsModule) 註冊到專案的 啟動模組(AppModule) 中。



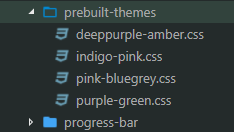
官方文件提出一個很好的元件管理方式，建立一個模組來集中管理有用到的元件，有元件需要使用時再將它加入到該元件所依附的模組內。

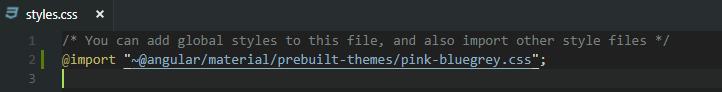
> ng g m customMaterial -–flat  
建立一個 CustomMaterialModule 使用—flat不產生資料夾，把檔案放在最外層的src\app底下。

## 匯入樣式

將安裝material後提供的css檔import至專案中的styles.css，路徑為@angular/material/prebuilt-themes/

擇一import至styles.css中





## 安裝手勢操作的套件 [HammerJS](http://hammerjs.github.io/)

使用Material時，在html中常會使用到mat開頭的tagName，例如: <mat-icon></mat-icon>，需要使用HammerJS連動這些tag給Material使用。

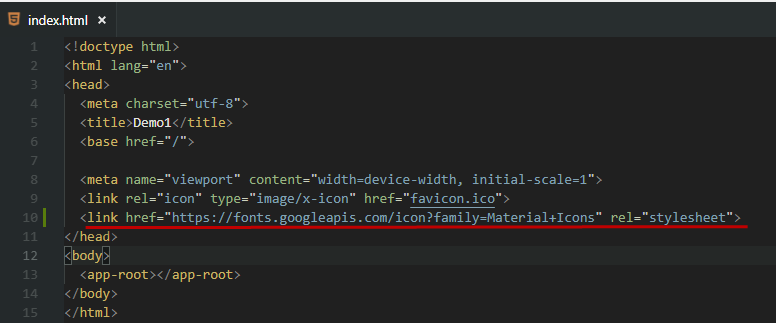
> npm install --save hammerjs

將 hammerjs import 到程式的**進入點**(entry point)，我們將下列語法加入至 src\main.ts。



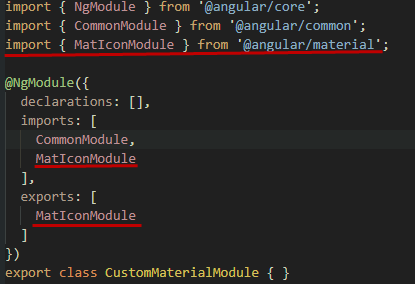
## 引入圖示套件 Material Icons

Angular Material 的圖示元件為 mat-icon，我們在首頁(src\index.html) 加入下面連結。  
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">



## 測試

打開 src\app\custom-material.module.ts 並將 **MatIconModule** 加到 imports 以及 exports 屬性內。



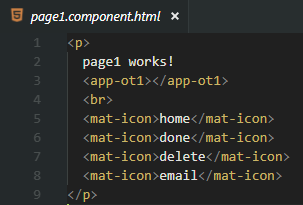
在 Page1Component 放入一些圖示：

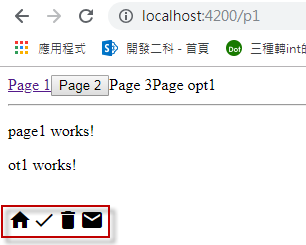
因為 Page1Component 使用 CustomMaterialModule 內的 MatIconModule，所以必須將 CustomMaterialModule 註冊到 Page1Component 所屬的模組(AppModule)內，開啟並編輯 src\app\app.module.ts。



在 src\app\page1\page1.component.html 內加入一些圖示，如下面範例：

開啟瀏覽器，可以看到已經出現圖示，更多圖式可去[官方文件](https://material.io/tools/icons/)中查找。



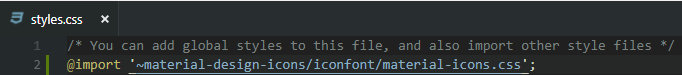


引入Material Icon也可以透過去[官方](https://google.github.io/material-design-icons/)下載css檔，並且放入專案中更改引入路徑。

其二，也可以透過npm安裝，

> npm install --save material-design-icons

然後開啟 styles.css 把css檔import進去



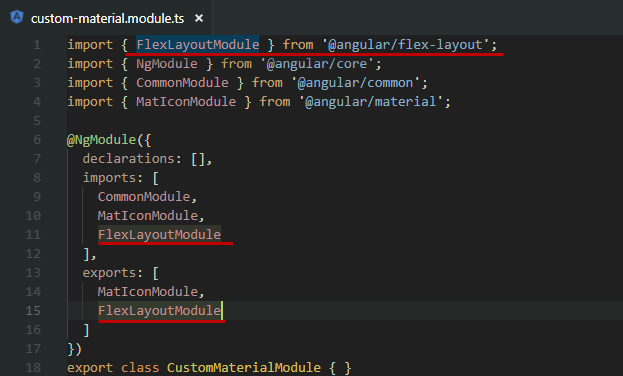
## Angular UI：Angular Flex-Layout

Angular Flex-Layout ([官方demo](https://tburleson-layouts-demos.firebaseapp.com/#/docs))實作了CSS3 flexbox排版，所以要注意流覽器是否支援該語法。

透過npm安裝

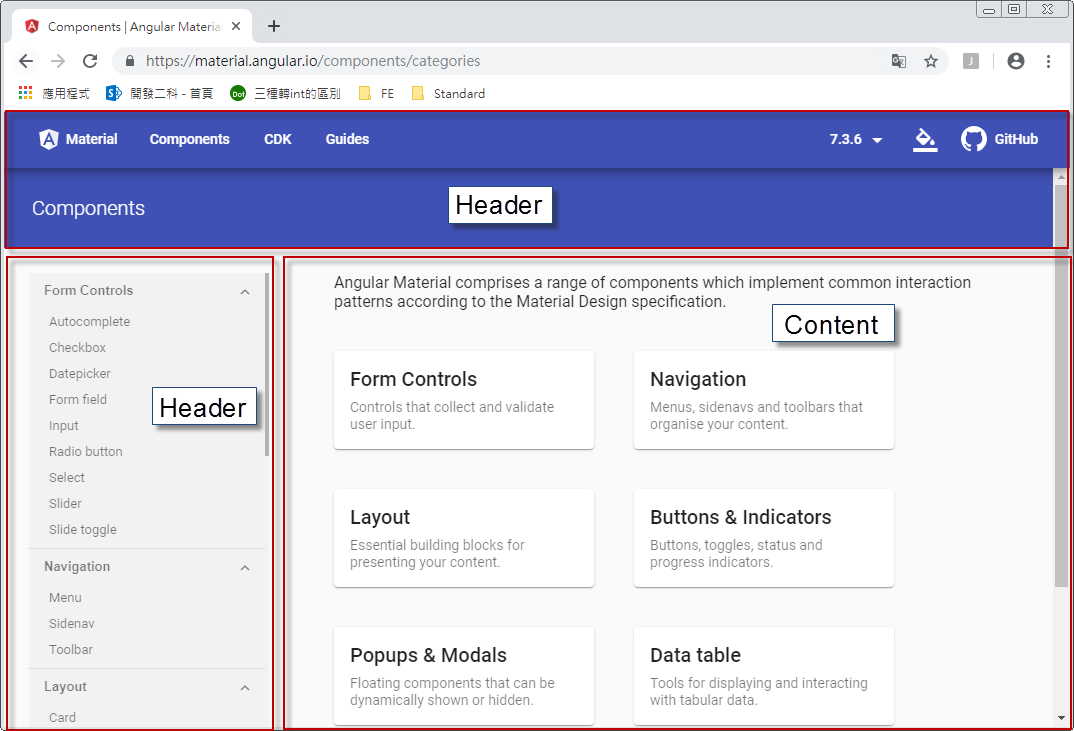
> npm install --save @angular/flex-layout@latest

一樣將此類(layout相關)Module放到custom-material.module.ts中，以便套用。



## 建立網站框架

嘗試建立一功能性網站，包含Header、側邊欄跟內容



建立一個HomeModule模組

> ng g m home –routing

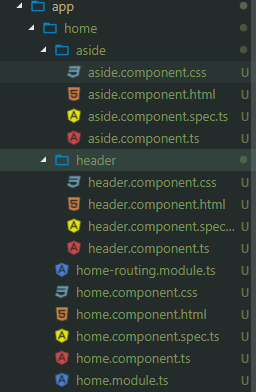
HomeModule下分別建立三Component => Home(不建立資料夾)、Header、Aside

> ng g c home\home --flat

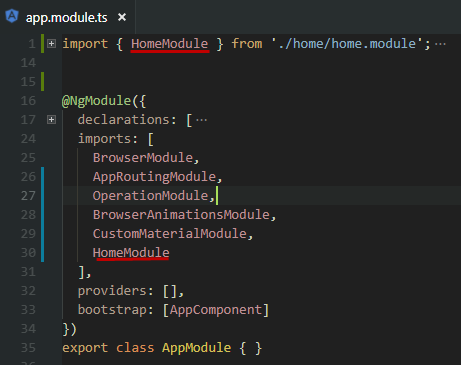
> ng g c home\header

> ng g c home\aside

此時，專案檔案應該會像這樣



接下來，一樣將HomeModule註冊到AppModule中，



修改 src\app\app-routing.module.ts，加入路由規則 home 對應到 **HomeComponent**，並將 空白的路由規則導引到 home路徑。



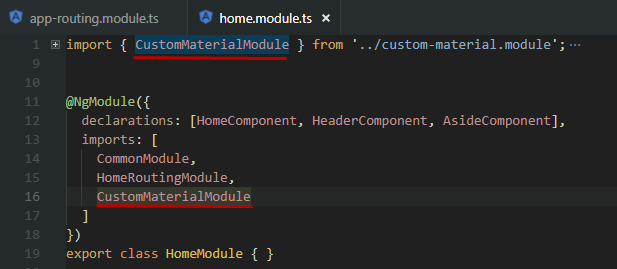
pathMatch：表示路由規則比對模式，full 表示須完全相同。

正常情況下如果路由規則是重新導向 (redirectTo)，都會加上 pathMatch: 'full' 的參數，萬用路由因為無法明確比對，所以可以不加，最簡單的方式就是一律加上去。

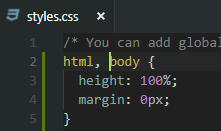
修改 src\app\app.component.html，只保留路由插座 (router-outlet)。

### 切版

因為HomeModule要使用到 Flex-Layout，所以將CustomMaterialModule註冊到HomeModule中。



修改 styles.css，做一些基礎設定

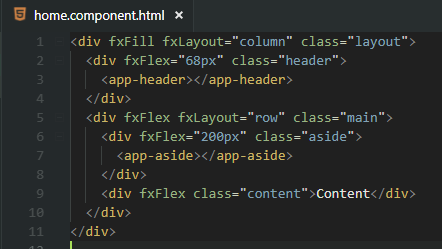


修改 src\app\home\home.component.html，將其切成3個區塊，並在css檔 src\app\home\home.component.css 內加入背景顏色以檢視版面是否正常。

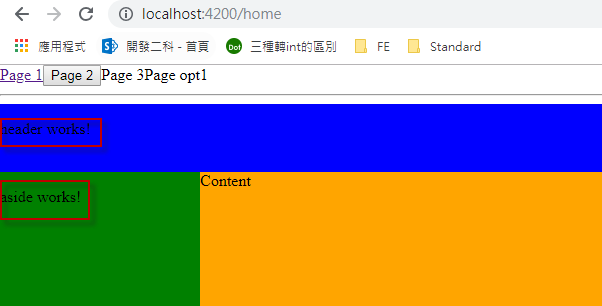
fxFill：fxFlexFill 的縮寫，表示填滿區域。

fxLayout：內容項目排列方式，row 表示優先以水平排列；colunm 表示優先以垂直排列。

fxFlex：有值時會以該值作為設定，無值時表示會佔用剩餘空間，其值會受到父元素的 fxLayout 屬性影響，當父元素為水平優先排列時(fxLayout='row')，fxFlex 會影響目前元素的寬度(width)；當父元素為垂直優先排列時(fxLayout='colunm')，fxFlex 會影響目前元素的高度(height)。



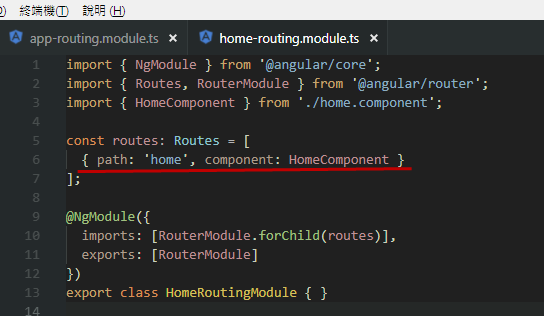
開啟瀏覽器，切板已完成，且header與aside也已套用進來。



嘗試比較AppRoutingModule與HomeRoutingModule，會發現其中RouterModule.forChild(routes)與RouterModule.forRoot(routes)的不同，差別在forRoot()為根路由，身為forChild()會以子路由的身分附加在根路由內。

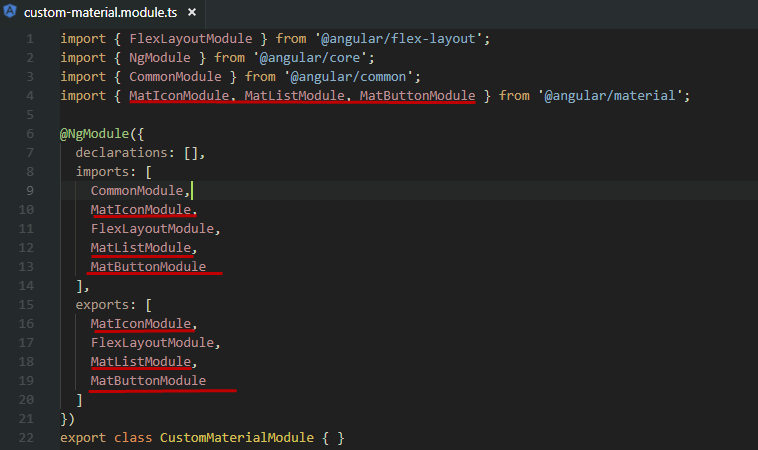
將AppModule內的路由規則 { path: 'home', component: HomeComponent } 移至 HomeRoutingModule。執行後會看到結果沒有任何改變，Angular 會幫我們將所有路由合併，並比對路由規則，也表示子路由加入的順序也會影響比對結果，只是正常情況下路由規格的條件(path 屬性)不應該重複。





### Aside區塊

在 Aside 區塊加入一些功能項目，因為會使用到 MatListModule、MatButtonModule、MatIconModule，所以開啟 src\app\custom-material.module.ts 將缺少的模組補上

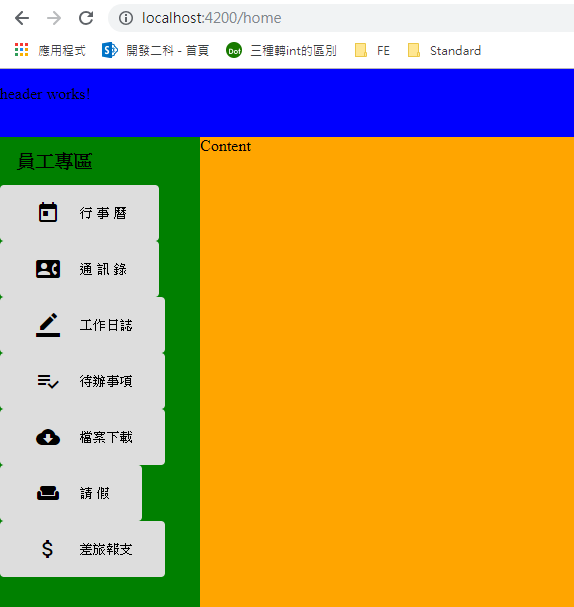


開啟 app\home\aside\aside.component.html 並修改html

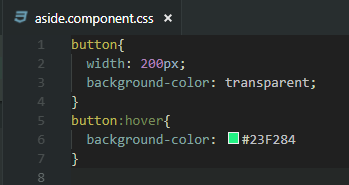
1. <mat-list>
2. <h3 mat-subheader>員工專區</h3>
3. <button mat-button routerLink="">
4. <mat-list-item>
5. <mat-icon mat-list-icon>today</mat-icon>
6. <h4 mat-line>行 事 曆</h4>
7. </mat-list-item>
8. </button>
9. <button mat-button routerLink="">
10. <mat-list-item>
11. <mat-icon mat-list-icon>contact\_phone</mat-icon>
12. <h4 mat-line>通 訊 錄</h4>
13. </mat-list-item>
14. </button>
15. <button mat-button routerLink="">
16. <mat-list-item>
17. <mat-icon mat-list-icon>border\_color</mat-icon>
18. <h4 mat-line>工作日誌</h4>
19. </mat-list-item>
20. </button>
21. <button mat-button routerLink="">
22. <mat-list-item>
23. <mat-icon mat-list-icon>playlist\_add\_check</mat-icon>
24. <h4 mat-line>待辦事項</h4>
25. </mat-list-item>
26. </button>
27. <button mat-button routerLink="">
28. <mat-list-item>
29. <mat-icon mat-list-icon>cloud\_download</mat-icon>
30. <h4 mat-line>檔案下載</h4>
31. </mat-list-item>
32. </button>
33. <button mat-button routerLink="">
34. <mat-list-item>
35. <mat-icon mat-list-icon>weekend</mat-icon>
36. <h4 mat-line>請 假</h4>
37. </mat-list-item>
38. </button>
39. <button mat-button routerLink="">
40. <mat-list-item>
41. <mat-icon mat-list-icon>attach\_money</mat-icon>
42. <h4 mat-line>差旅報支</h4>
43. </mat-list-item>
44. </button>
45. </mat-list>

關於 <mat>系列tagName用法請參考[Angular官方文件](https://material.angular.io/components/autocomplete/overview)

開啟瀏覽器，側邊欄已完成。



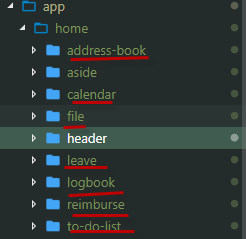
修改aside.component.css 讓button長得好看一點。



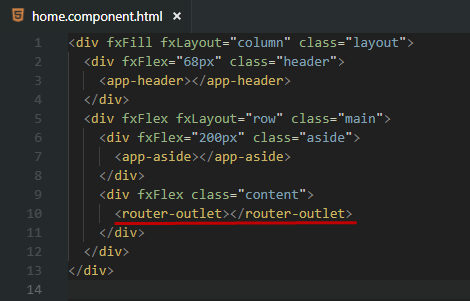


每個側邊欄的欄位都有各自的內容需要顯示，所以現在要新增每一個的各自的Component，

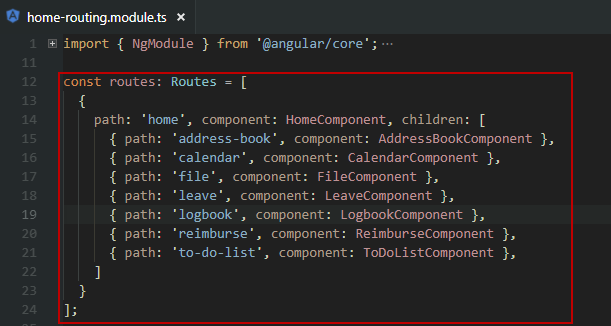
> ng g c home\calendar  
> ng g c home\addressBook  
> ng g c home\logbook  
> ng g c home\toDoList  
> ng g c home\file  
> ng g c home\leave  
> ng g c home\reimburse



開啟 src\app\home\home.component.html，在 Content 區塊內插入第二層的 router-outlet。



開啟 src\app\home\home-routing.module.ts，依照剛所產生的元件建立對應的路由規則，比較不同的是這次的路由規則是建立在 home 這個路由規則的 children 屬性內。



上方建立的route規則為，需要符合所設的路由規則路徑必須包含home，必須先滿足path:'home'，這條路由規則才會繼續比對children內的路由規則

開啟 src\app\home\aside\aside.component.html 並在各功能選單加入 routerLink

<mat-list>

<h3 mat-subheader>員工專區</h3>

<button mat-button routerLink="./calendar">

<mat-list-item>

<mat-icon mat-list-icon>today</mat-icon>

<h4 mat-line>行 事 曆</h4>

</mat-list-item>

</button>

<button mat-button routerLink="./address-book">

<mat-list-item>

<mat-icon mat-list-icon>contact\_phone</mat-icon>

<h4 mat-line>通 訊 錄</h4>

</mat-list-item>

</button>

<button mat-button routerLink="./logbook">

<mat-list-item>

<mat-icon mat-list-icon>border\_color</mat-icon>

<h4 mat-line>工作日誌</h4>

</mat-list-item>

</button>

<button mat-button routerLink="./to-do-list">

<mat-list-item>

<mat-icon mat-list-icon>playlist\_add\_check</mat-icon>

<h4 mat-line>待辦事項</h4>

</mat-list-item>

</button>

<button mat-button routerLink="./file">

<mat-list-item>

<mat-icon mat-list-icon>cloud\_download</mat-icon>

<h4 mat-line>檔案下載</h4>

</mat-list-item>

</button>

<button mat-button routerLink="./leave">

<mat-list-item>

<mat-icon mat-list-icon>weekend</mat-icon>

<h4 mat-line>請 假</h4>

</mat-list-item>

</button>

<button mat-button routerLink="./reimburse">

<mat-list-item>

<mat-icon mat-list-icon>attach\_money</mat-icon>

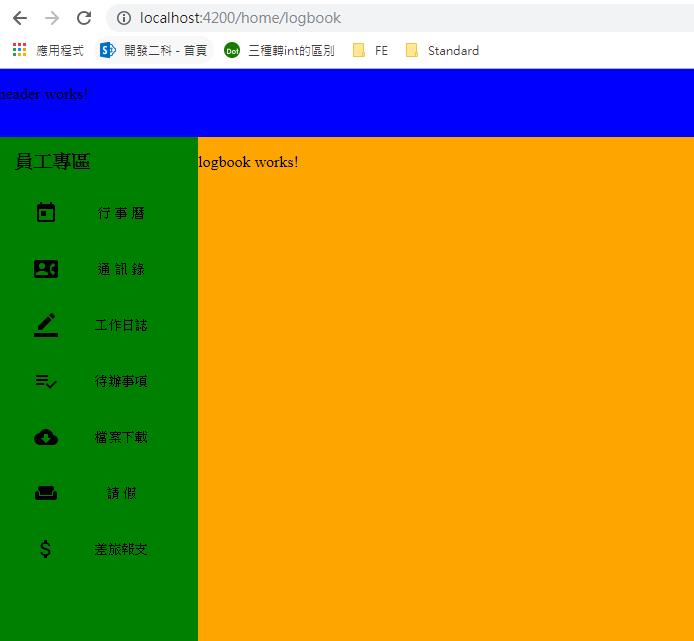
<h4 mat-line>差旅報支</h4>

</mat-list-item>

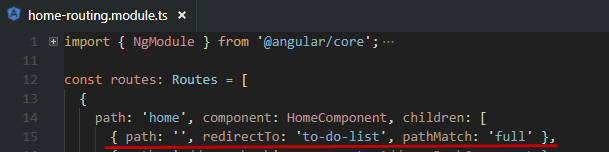
</button>

</mat-list>

開啟瀏覽器後，點擊側邊欄，就可以對應到各自的html



預設情況下Content 區塊會是空白，可以設定某個功能為預設功能，我們以待辦事項當作預設功能為例。

打開home-routing.module.ts 在routes裡面修改預設路徑，

## Angular 模組化

在 Angular 模組 我們練習先建立一個模組(NgModule)，然後在上面建立元件(Component)，今天我們試著反過來將既有的元件包覆到一個模組內。

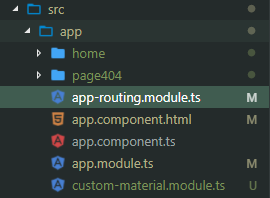
## 重整

先將用不到的元件及模組刪除：PagexComponent、OptxComponent、OperationModule，直接刪除檔案，再將把註冊相關元件的模組(src\app\app.module.ts、src\app\app-routing.module.ts)程式碼給移除。

可以利用指令 ng lint，查看目前專案中哪些檔案有出錯。

相關的樣板檔(app.component.css)以及測試檔(app.component.spec.ts)可以先移除，後續有需要時再加回來。

整理完成後



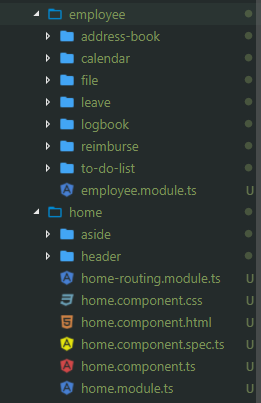
## 建立模組

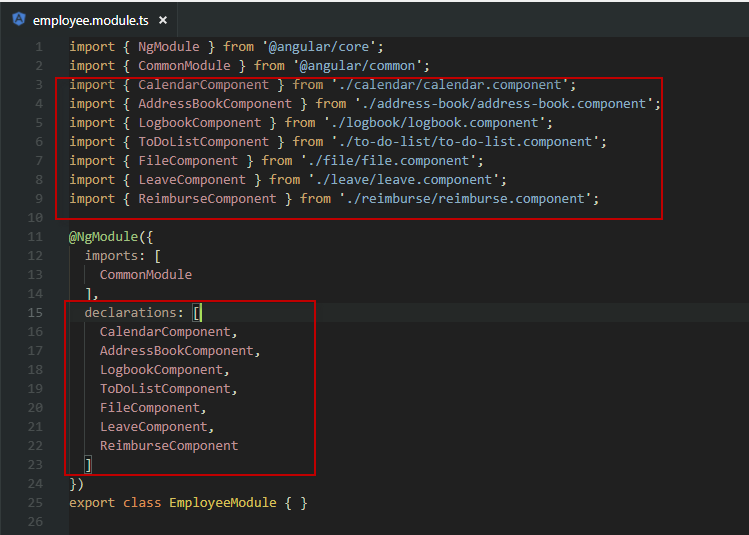
開始將在 Angular 子路由 所建立的功能元件都合併到一個模組內。

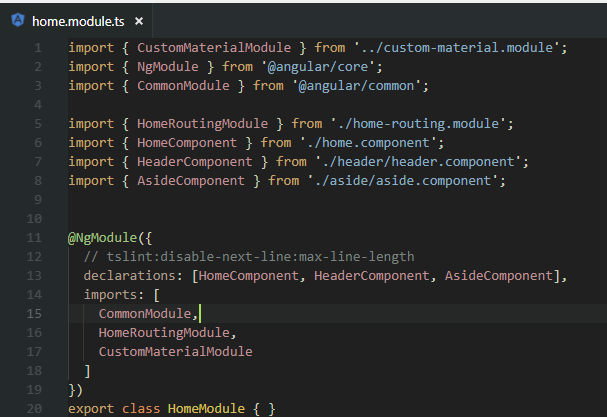
透過 ng g m employee 建立一個 EmployeeModule 模組。

> ng g m employee

將相關模組移動到 src\app\employee 下面，在可以透過 VS Code 的檔案總管用滑鼠直接拖拉，並將相關元件註冊到 src\app\employee\employee.module.ts。



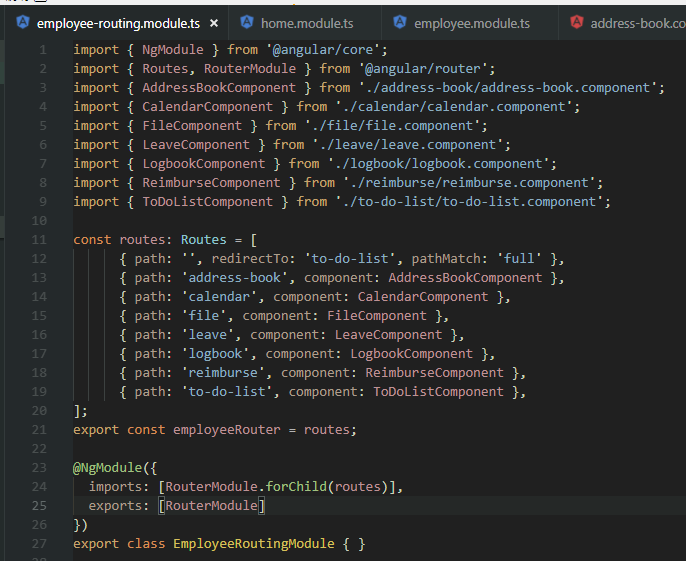




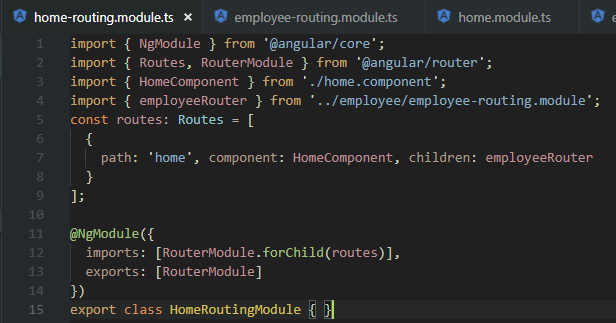
最後我們要處理的是HomeRoutingModule(src\app\home\home-routing.module.ts) 的路由規則，

將原本在Home router的路由規則移到 Employee 資料夾中，新增存放路由規則的ts檔，最後再把路由規則export出來給Home router使用

新增的 employee-routing.module.ts



原本的 home-routing.module.ts



開啟瀏覽器結果，會跟原本的效果一樣，但是會更加好管理。

資料綁定

資料繫結的四種方法 (範本語法 )

**內嵌繫結**( interpolation )

{{statement}}

•**屬性繫結**( Property Binding )

[propertyName]="statement"

[attr.attributeName]="statement"

•**事件繫結**( Event Binding )

(eventName)="someMethod($event)"

(eventName.key)="someMethod($event)"

•**雙向繫結**( Two-way Binding )

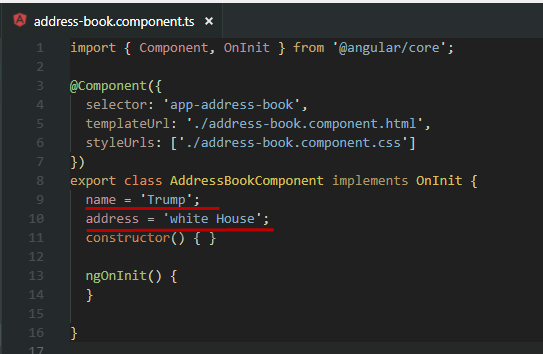
[(ngModel)]='property'

內嵌繫結( interpolation )

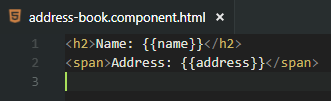
使用專案的address-book Component做為示範，內嵌繫結是透過 ()小括號

打開 address-book.component.ts

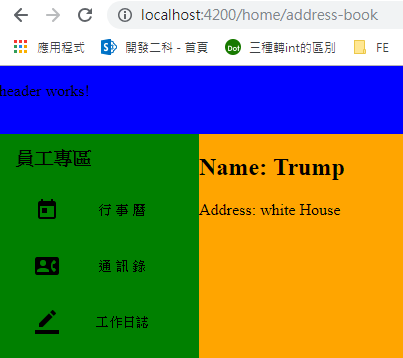
在exports程式碼內，宣告二變數為 ‘name’、’address



打開 address-book.component.html，修改程式碼



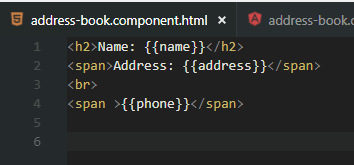
開啟瀏覽器觀察結果，可以發現我們在address-book.component.ts，宣告的變數‘name’、’address 成功繫結並顯示在畫面中，Angular可以透過 {{ }}將宣告的變數直接顯示在html中。[補充介紹](https://ithelp.ithome.com.tw/articles/10204430)

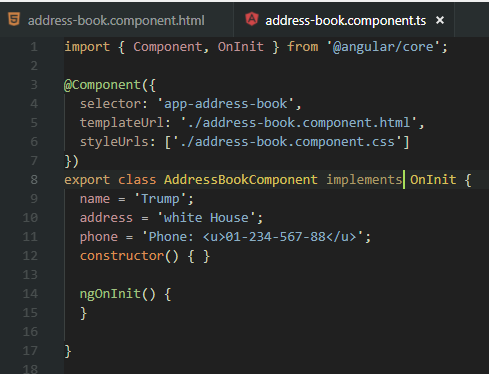


屬性繫結( Property Binding )

直接使用範例演練，屬性繫結是透過 []中括號

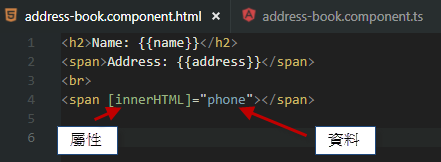
打開 address-book.component.html、address-book.component.ts，修改程式碼





其中看到phone 變數含有<u> html標籤，打開瀏覽器查看結果，畫面中直接把這html標籤顯示出來，所以這時候就要透過屬性標籤，將這html標籤綁定在這span 原有的 innerHTML 屬性。





打開瀏覽器查看結果，就可以看到顯示正確了。

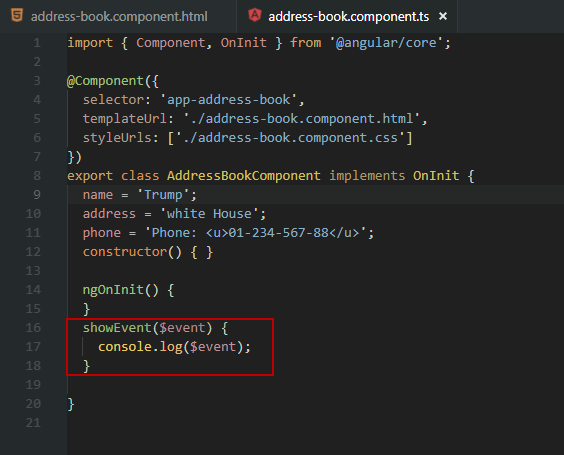


[補充介紹](https://ithelp.ithome.com.tw/articles/10207709)

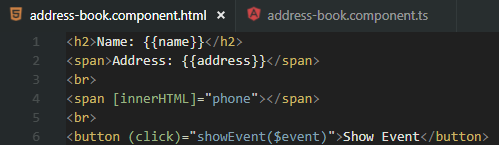
事件繫結( Event Binding )

使用事件繫結來得知按鈕按下的事件。

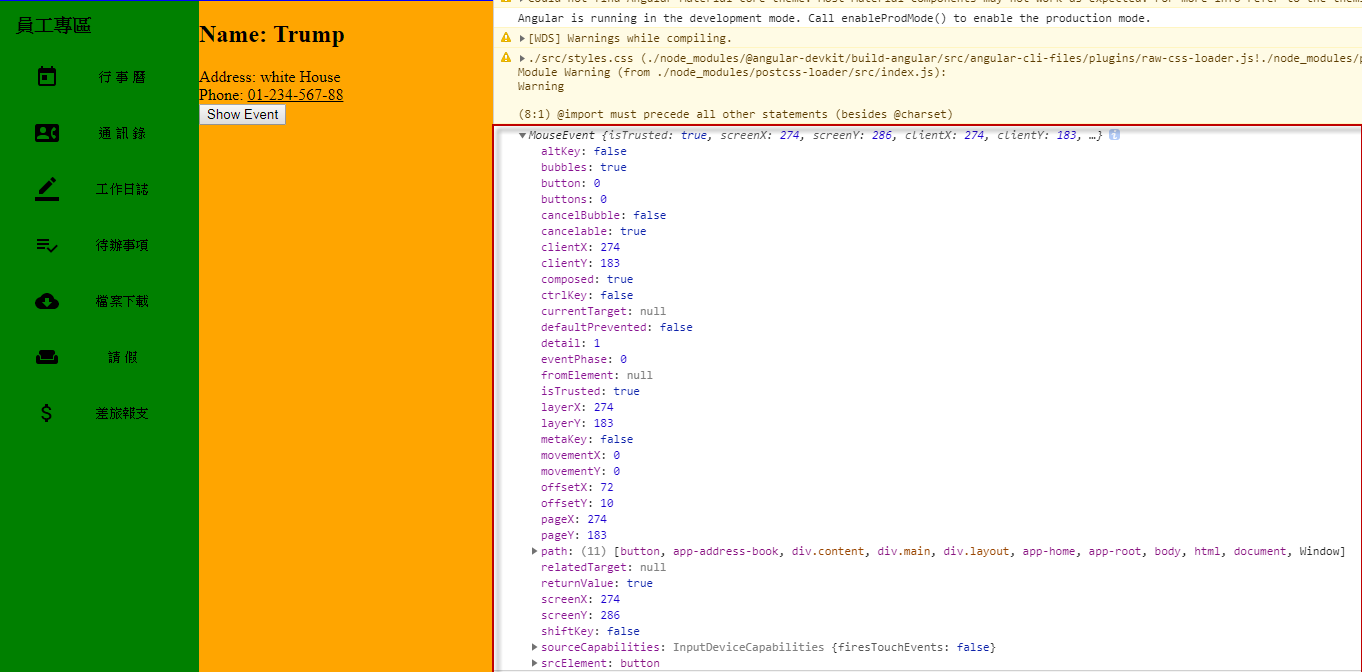
在address-book.component.ts加入showEvent()方法，console.log事件。



打開address-book.component.html修改程式碼，新增按鈕綁定showEvent()方法



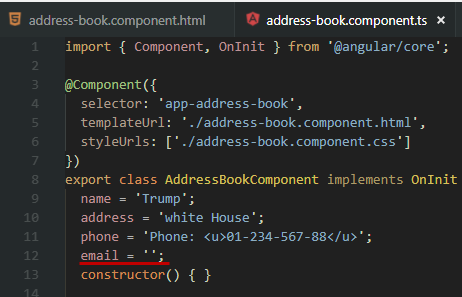
打開瀏覽器，查看執行結果點擊剛剛新增的Show Event按鈕，可以發現console視窗顯示此 click 事件其中的property



雙向繫結( Two-Way Binding )

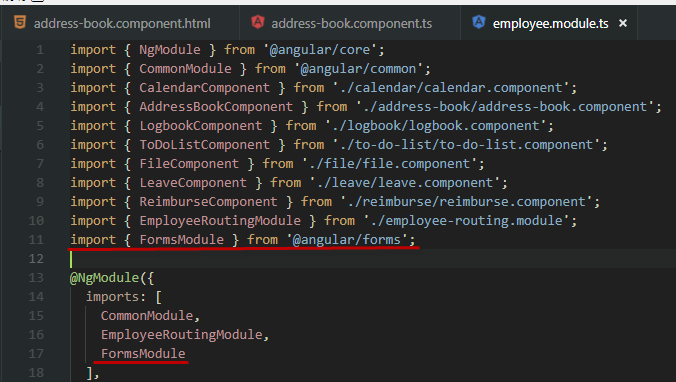
在輸入框中使用雙向繫結綁定資料，並將綁定的資料顯示在該欄位。

在address-book.component.ts宣告email變數，並且給予空字串，因為透過雙向繫結將動態修改此變數。



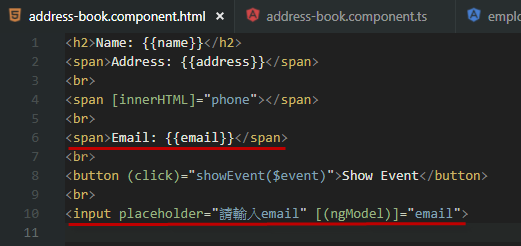
並且在此module中註冊FormsModule，其中成員ngModel將會被使用到。

打開AddressBook Component的所屬模組 Employee.module.ts註冊FormsModule



打開address-book.component.html修改程式碼，新增一個input標籤做雙向綁定和一個span標籤存放輸入訊息

請注意，雙向繫結在於 [(ngModel)] = “宣告變數”。



打開瀏覽器，在input輸入窗輸入文字，會將此輸入文字動態顯示在上方span標籤中。



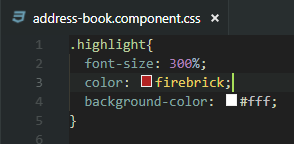
雙向繫結就是**屬性繫結+事件繫結**

[ngModel]+(ngModelChange)=[(ngModel)]

**屬性指令**

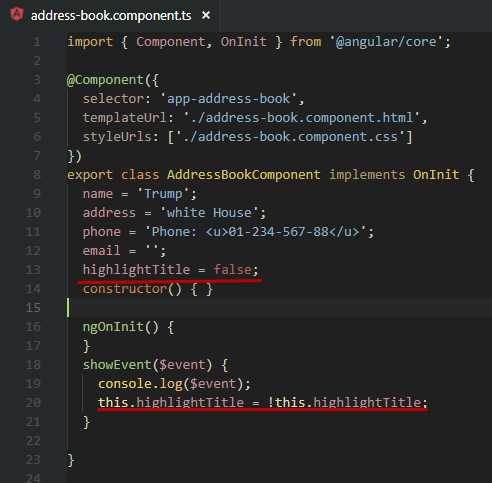
透過屬性指令在按下按鈕時切換文字樣式。

打開 address-book.component.css加入css樣式



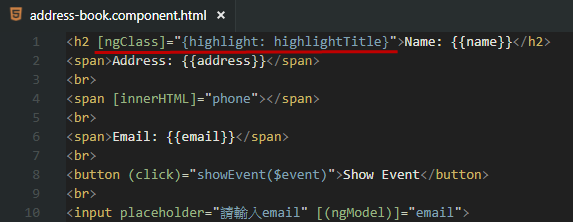
打開 address-book.component.ts加入hightlightTitle屬性切換樣式

請注意，其中this.變數名稱 表示是此Component的變數，!表示反向，所以代表點擊之後false => true、true => false

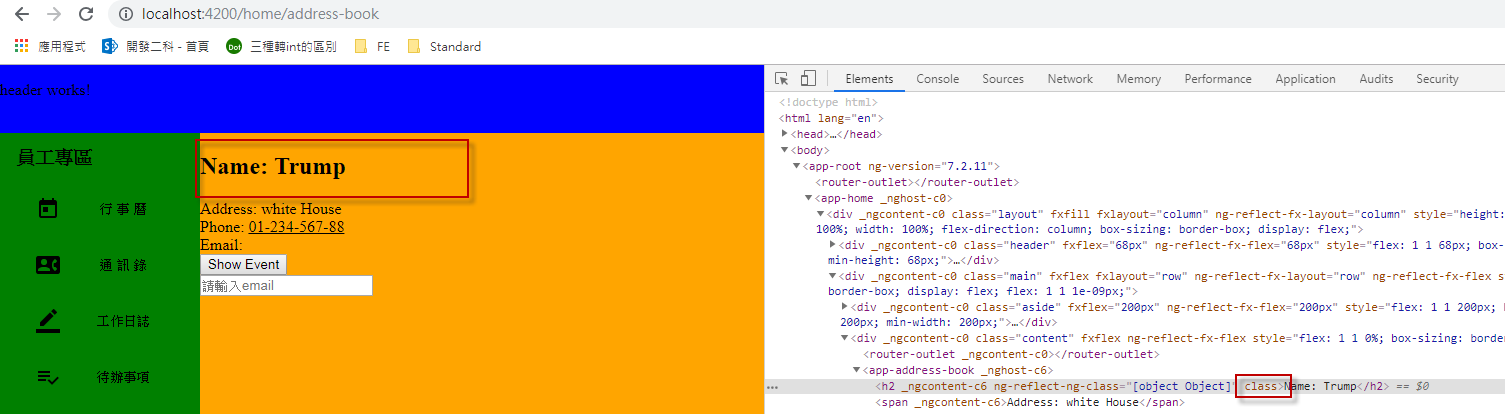


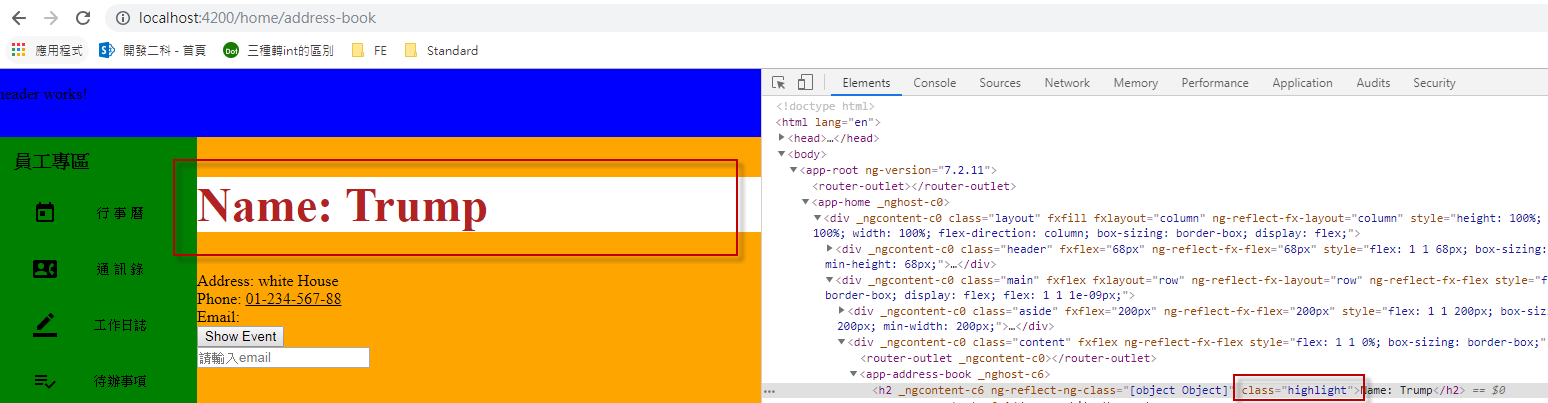
打開 address-book.component.html加入hightlightTitle屬性切換樣式，

其中ngClass代表angular綁定class的方式，而highlight表示須綁定的class，highlightTitle表示是否該綁定(true or false)



打開瀏覽器並點擊 Show Event 按鈕查看畫面結果，可以看到多次點擊 其中 <h2> Trump欄位，會切換選擇是否擁有 class highlight如下圖顯示





**結構指令 \*ngIf,\*ngFor**

當未輸入查詢文字時，使用 \*ngIf 提示使用者。

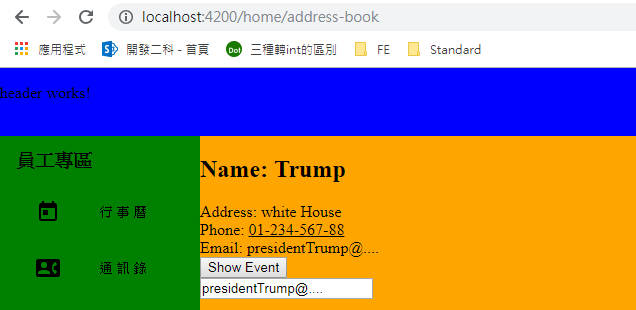
在 address-book.component.html中加入提示文字，並加上\*ngIf設定顯示條件。

\*ngIf 將會判斷引號中條件是否成立，成立(true)就顯示，反之就不顯示



打開瀏覽器並查看結果。



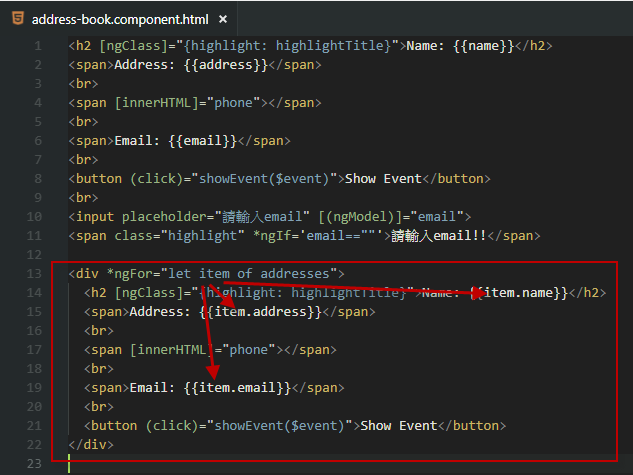


使用\*ngFor顯示陣列資料，透過陣列數量去顯示應有的項目。

打開 address-book.component.ts加入一陣列並含有聯絡物件



打開 address-book.component.html並加入程式碼，\*ngFor 變數中addresses 即為剛剛宣告之陣列，Addresses -> item，item即擁有addresses內物件之資訊



打開瀏覽器並查看顯示結果，紅框內即為addresses內物件



**@Input()傳遞資料**

Angular父、子元件如果需要資料的傳遞，例如: 父元件的樣板(html)需要顯示子元件的資料，就會需要 @Input()、@Output() 接收、傳遞資料。

這邊首先演練 @Input()

新增一個Component 在app Module底下，

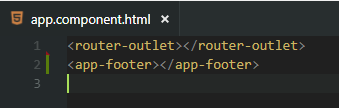
> ng g c footer

如果遇見底下這個情況，使用



> ng g c footer –-module app強迫建立元件在app Module內

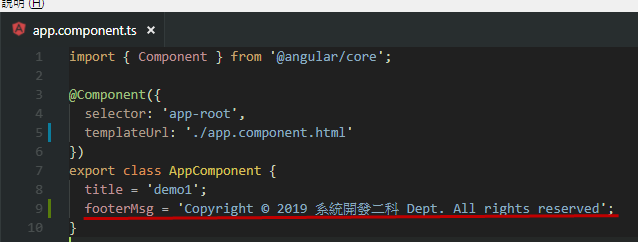
打開 app.component.html 在<router-outlet></router-outlet> 將剛剛新增的 footer Component放進去



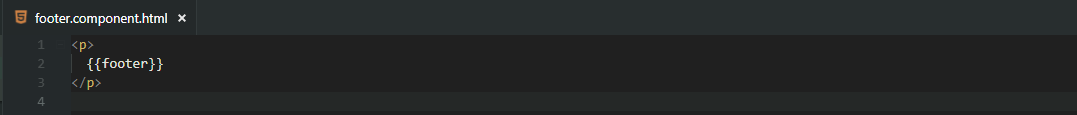
打開瀏覽器，會發現畫面下面已經多了 footer



打開 app.component.ts 宣告要在footer中顯示的字樣，



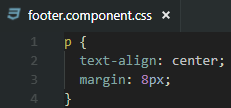
打開 footer.component.ts 宣告(@Input)要在footer中顯示的字樣，此處的footer即將從app.component.ts 宣告的footerMsg傳遞進來並顯示在footer.component.html(等等即將新增)

修改 footer.component.html 修改顯示內容

打開 app.component.html ，將app裡面的footerMsg指至footer元件裡面的footer變數



打開 footer.component.css，修改css樣式



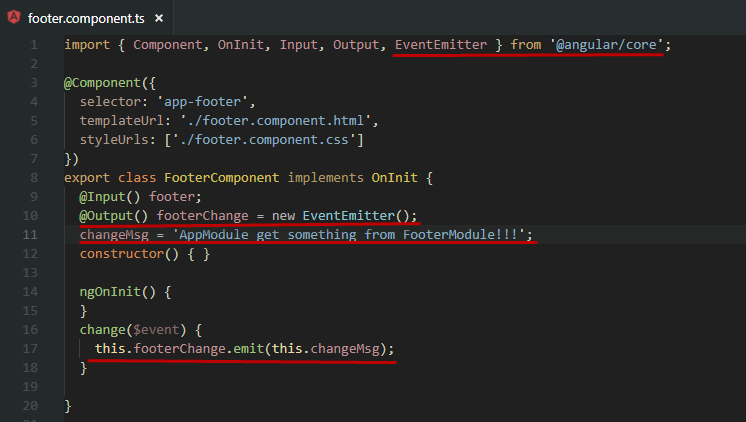
打開瀏覽器查看結果，看到父元件AppComponent的變數已經傳遞到子元件FooterComponent



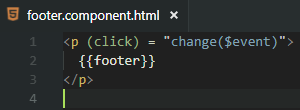
**@Output()傳遞資料**

接下來演練子元件傳遞資料給父元件使用

首先打開 footer.component.ts 宣告需要傳遞的變數 changeMsg ，接著使用 @Output 建立一個發射器 EventEmitter()，這個 發射器 EventEmitter() 是Angular 原有的方法，所以import的地方也是從@angualr/core，接著新增一個方法，這個方法做的是 讓這個發射器EventEmitter() 把需要發射出去的東西(changeMsg)發射(emit)出去。

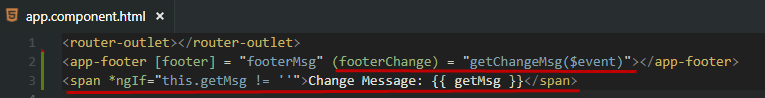


然後在 footer.component.html 綁定click事件在剛剛新增的方法



接著打開 app.component.html，加上程式碼接受 子元件 footerChange 發射器 發射出來的東西，getChangeMsg($event) 為AppComponent 本身的方法，等等將會說明。

新增<span>標籤是為了顯示接受的資訊的顯示欄位。



打開 app.component.ts，新增存放接收資訊的變數及實作方法。透過實作方法 getChangeMsg 將接收到的資訊指給 變數getMsg



打開瀏覽器並實作點擊footer動作，將footer原有變數發射至給AppModule使用，查看結果。



**自訂服務元件(Services)**

將AppModule變數資訊或是方法移動到一個 Service 統一管理，需要使用時再將服務進行注入。

> ng g s appServices 新增一個services，s代表 Services的縮寫



把 app.component.ts 內宣告的變數搬移到 AppServicesService 中



接著在 app.component.ts 中把AppServicesService的 內容注入進來，並宣告變數接收 AppServicesService，並且透過建構式把AppServicesService 注入到appComponent的變數中



打開瀏覽器並檢視結果，發現與原本內容並無不同，但式已經本來在AppComponent的變數移到services統一管理，如果有其他的Component的內容，也可以這樣管理。



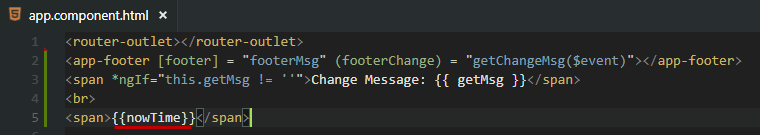
**Pipe元件**

Angular Pipe 是一個強大的資料顯示轉型的工具，可以透過Pipe 的幫忙，將原始資料轉換成我們想要顯示的樣式，且不會異動到原始資料的內容。

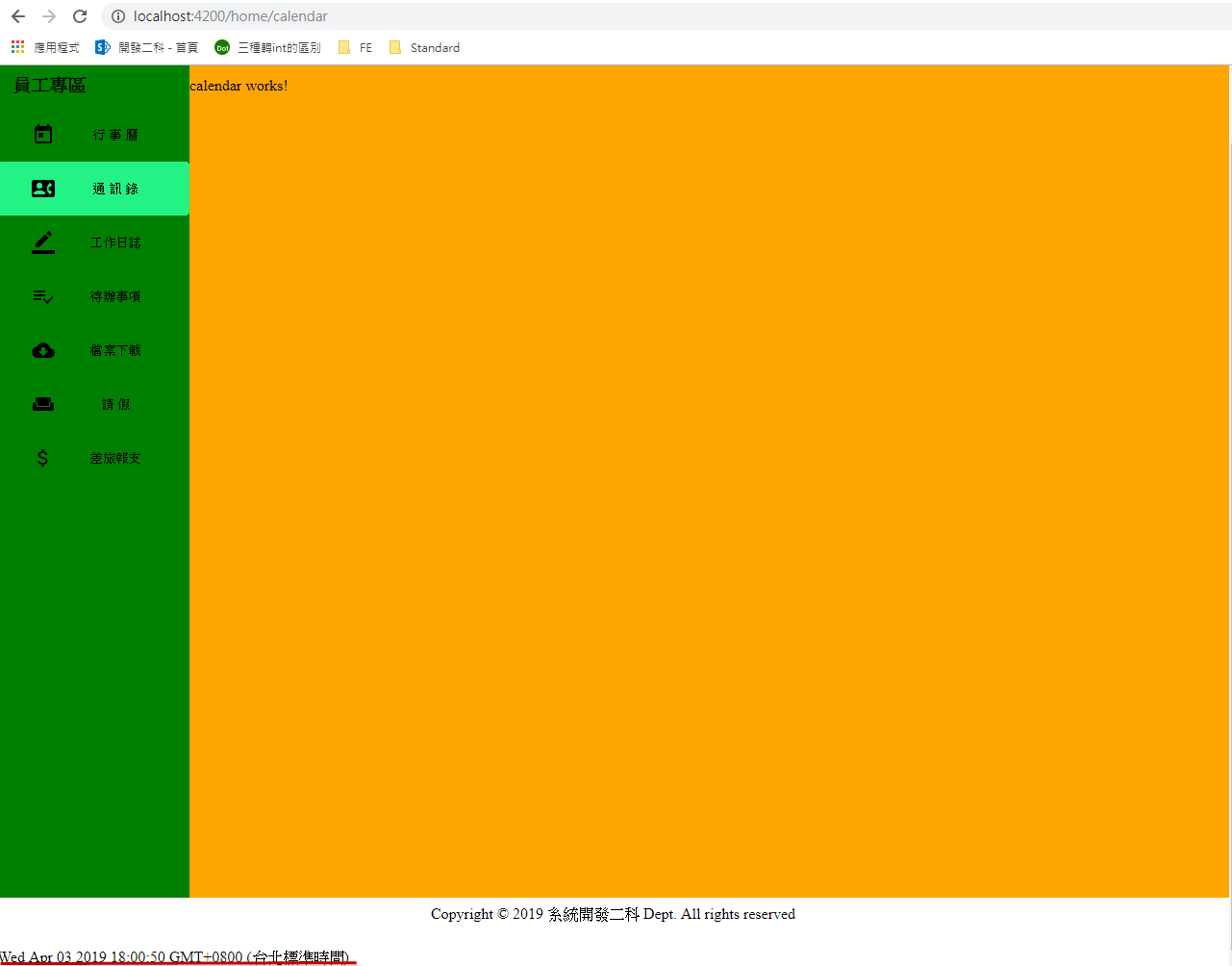
使用 DatePipe 來改變日期格式

在 app.component.ts 中 宣告存放時間的變數 nowTime = Date()，Date()表示現在時間，接著在app.component.html中透過內嵌繫結顯示此變數。

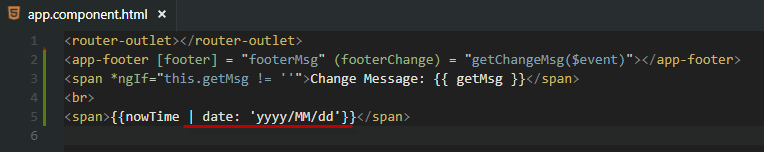




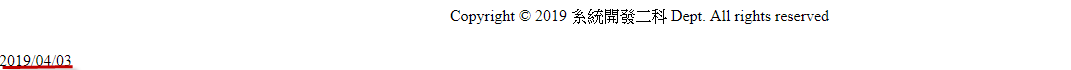
打開瀏覽器查看，看到畫面最下方有剛剛加入的時間。



接著透過 DataPipe改變顯示方式，DatePipe 更多使用方式請參考[官方文件](https://angular.io/api/common/DatePipe)



接著打開瀏覽器查看 時間日期 已經變為顯示成 YYYY/MM/DD



Angular 除了 DatePipe 之外，很貼心地提供了非常多類似的 Pipe 如：

* 處理貨幣單位非常方便的 [CurrencyPipe](https://angular.io/api/common/CurrencyPipe)
* 自動 JSON 格式化的 [JsonPipe](https://angular.io/api/common/JsonPipe)
* 幫你自動訂閱的 [AsyncPipe](https://angular.io/api/common/AsyncPipe)
* 英文大小寫處理的 [UpperCasePipe](https://angular.io/api/common/UpperCasePipe)、[LowerCasePipe](https://angular.io/api/common/LowerCasePipe) 與 [TitleCasePipe](https://angular.io/api/common/TitleCasePipe)
* 專門處理小數的 [DecimalPipe](https://angular.io/api/common/DecimalPipe)
* 小數點轉百分比很方便的 [PercentPipe](https://angular.io/api/common/PercentPipe)
* 陣列跟字串都可以用的 [SlicePipe](https://angular.io/api/common/SlicePipe)
* 多國語系處理專門的 [I18nSelectPipe](https://angular.io/api/common/I18nSelectPipe) 與 [I18nPluralPipe](https://angular.io/api/common/I18nPluralPipe)
* 暫時想不到用途的 [KeyValuePipe](https://angular.io/api/common/KeyValuePipe)

其他相關說明介紹可參考：https://ithelp.ithome.com.tw/articles/10204799