



Angular 13 開發實戰 新手實作篇

(解答本)

多奇數位創意有限公司

全端工程師 黃升煌 (Mike)

<https://fullstackladder.dev>





開發環境準備

準備開發環境

- 參考 [Angular 13 開發環境說明](#) 進行安裝設定
 - 安裝 Git 版控工具
 - 安裝 Node.js v16.13.0 LTS 以上版本
 - 確認 npm 為 v8.1.0 以上版本
 - 確認 Angular CLI 為 v13.0.0 以上版本
 - 安裝 Visual Studio Code (請更新至最新版)
 - 安裝 [Angular Extension Pack](#) 擴充套件

程式碼相關

- 使用 RealWorld 樣板
 - <https://github.com/gothinkster/realworld>
- 針對課程調整過的樣板
 - <https://github.com/coolrare/realworld-basic-template>
- 完整程式碼及開發過程
 - <https://github.com/coolrare/angular-realworld-basic>

安裝 Angular Extension Pack

The screenshot shows the Visual Studio Code Marketplace interface. On the left, there's a sidebar with various icons. In the main area, a search bar at the top has the text '擴充功能: 市集'. Below it, a list of extensions is displayed:

- Angular Extension Pack 0.5.2** by Loiane Groner (23K installs, 5 stars). A red circle with the number '2' is placed over this item.
- Angular Extension Pack 1.3.2** by Will 保哥 (105K installs, 4.5 stars). A red circle with the number '3' is placed over this item.
- Angular 6 Snippets - Typ...** by Mikael Morlund (2.6M installs, 5 stars).
- Angular v5 Snippets** by John Papa (2.8M installs, 5 stars).
- Angular Extension Pack 0.0.4** by Plínio Naves (713 installs, 5 stars).
- Angular Extension Pack 0.0.13** by rexebin (107 installs).
- Angular Extension Pack 1.0.0** (888 installs).

On the right, a detailed view of the first extension is shown:

Angular Extension Pack by doggy8088.angular-extension-pack (105,942 installs, 5 stars)

Popular Visual Studio Code extensions for Angular Development

安裝 (Install button)

安裝完成後記得重新啟動 Visual Studio Code (Text overlay in red)

Details: 詳細資料 | 貢獻 | 變更記錄 | 相依性

Angular Extension Pack

This extension pack packages some of the most popular (and some of my favorite) Angular extensions. If you like it, please [Review](#) and share with your friends. If you know any extension that is good for Angular development, just let me know by [posting a comment](#).

Extensions Included

- Angular Code Snippets



主軸1：Angular 專案基礎架構

任務01：建立專案基礎架構

- 請使用 `ng new` 指令，建立一個全新的 Angular 專案



任務01：建立專案基礎架構

- `ng new realworld-basic`
- `cd realworld-basic`
- `npm start` 或 `ng serve`



補充：使用 yarn 作為套件管理器

- `ng set --global packageManager=yarn`

任務02：現有模板切版，及元件規劃

- 下載模板

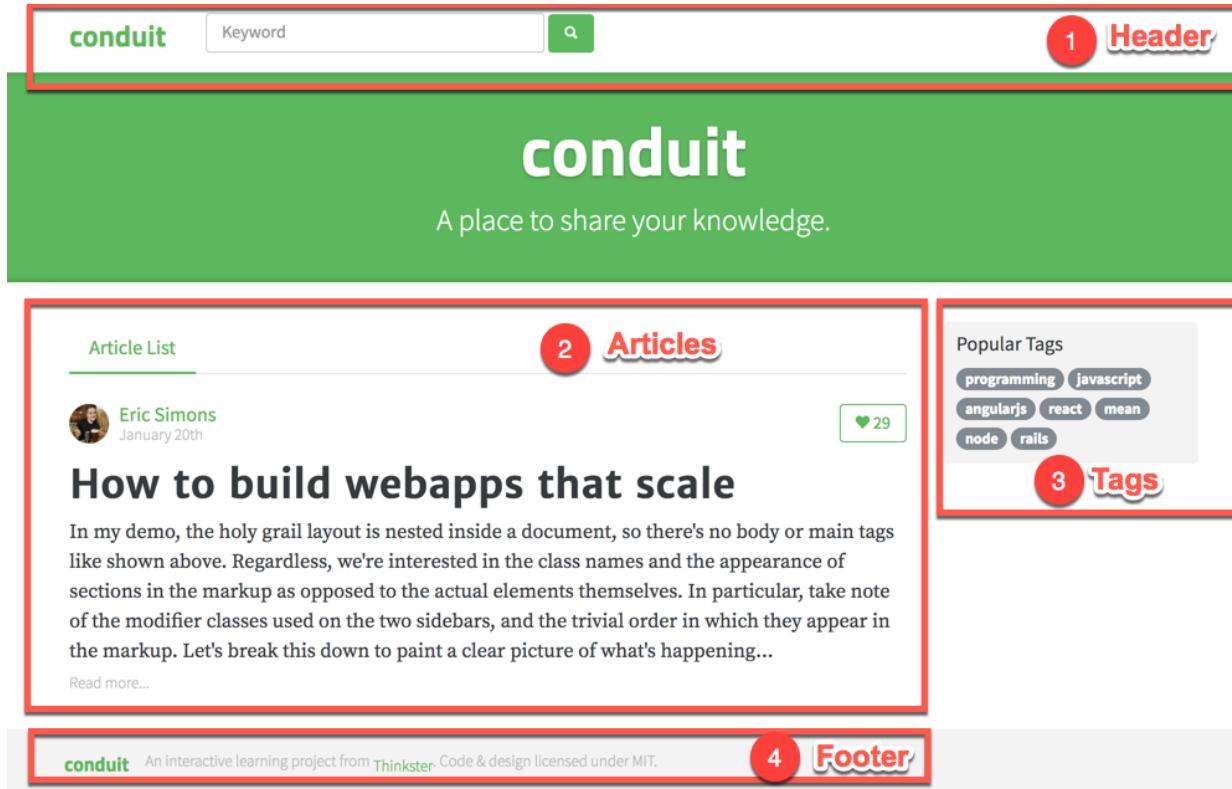
The screenshot shows a GitHub repository page for 'coolrare / realworld-basic-template'. The page includes navigation links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Security, Insights, and Settings. It displays a message: 'No description, website, or topics provided.' with an 'Edit' button. Below this, it shows 5 commits, 1 branch, 0 releases, and 1 contributor. A dropdown menu for the master branch is open. At the bottom, there's a list of files: assets (Update style.css) and index.html (Remove unnecessary content). A callout box highlights the 'Clone with HTTPS' section, which contains the URL <https://github.com/coolrare/realworlc> and a copy icon. Below this are 'Open in Desktop' and 'Download ZIP' buttons, with 'Download ZIP' being highlighted by a red rectangle.

任務02：現有模板切版，及元件規劃

- 將 assets 目錄內容搬到專案的 src/assets 中
- 將 index.html 內容搬到 src/index.html
- index.html 的 <head></head> 區塊中
 - 加入 <base href="/">
- 將 index.html 內容的 body 區塊
 - 剪下移到 app.component.html
- 在 index.html 的 body 區塊中
 - 加上 <app-root></app-root>

任務02：現有模板切版，及元件規劃

- 將 `app.component.html` 的內容，依照下圖指示切割成 4 個元件



任務02：現有模板切版，及元件規劃

- 建立 header component
 - ng g c header

```
wellwind □ /Users/wellwind/GitHub/realworld-basic □ mission-02 ✓ □ ng g c header
CREATE src/app/header/header.component.css (0 bytes)
CREATE src/app/header/header.component.html (25 bytes)
CREATE src/app/header/header.component.spec.ts (628 bytes)
CREATE src/app/header/header.component.ts (269 bytes)
UPDATE src/app/app.module.ts (396 bytes)
```

任務02：現有模板切版，及元件規劃

- 將 header 部分的 html 搬到 header.component.html

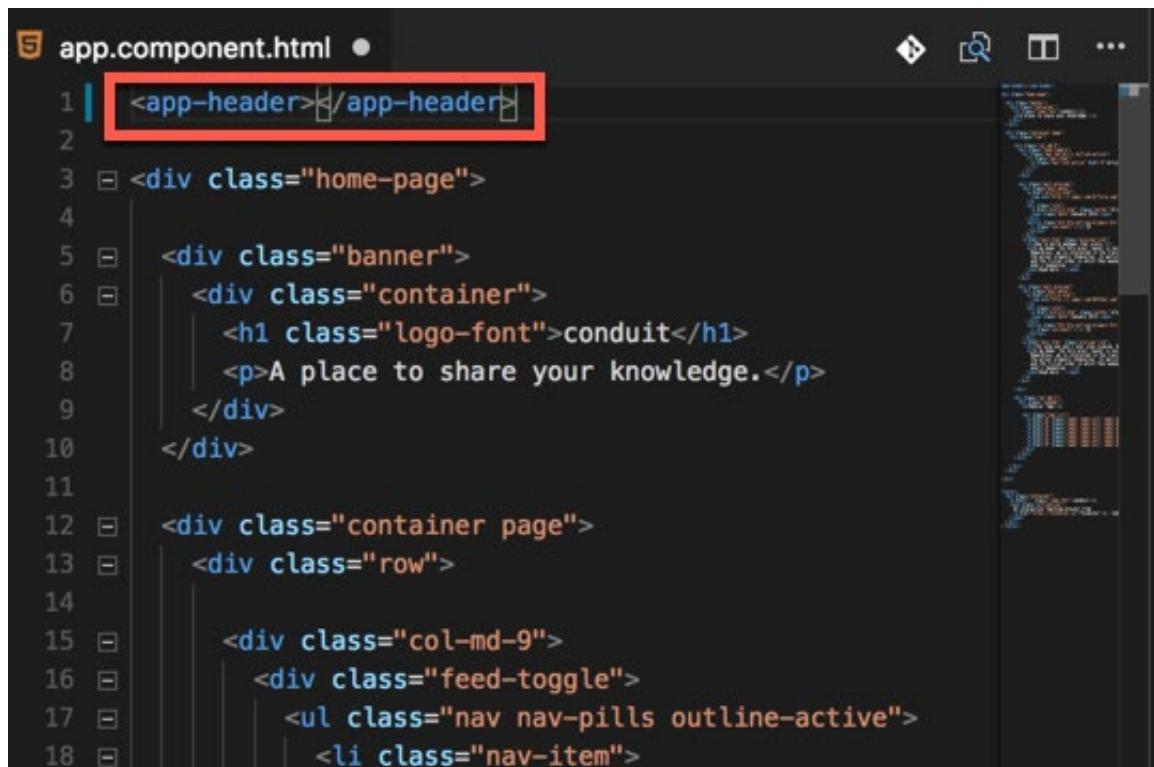
The screenshot shows a code editor interface with three main panes. On the left is a file tree titled '檔案總管' (File Explorer) showing a project structure under 'REALWORLD-BASIC'. The 'src/app/header' folder contains four files: 'header.component.css', 'header.component.html', 'header.component.spec.ts', and 'header.component.ts'. The 'header.component.html' file is currently open in the center pane. A red box highlights the first nine lines of the code, which define a navigation bar (navbar navbar-light) with a brand link, a search input, and a search button. An orange arrow points from this highlighted area towards the right pane, where the 'header.component.html' file is also visible but appears to be empty or has been moved.

```
<nav class="navbar navbar-light">
  <div class="container">
    <a class="navbar-brand" href="index.html">conduit</a>
    <input class="form-control form-control-sm search-in">
    <button class="btn btn-sm btn-primary">
      <i class="ion-search"></i>
    </button>
  </div>
</nav>

<div class="home-page">
  <div class="banner">
    <div class="container">
      <h1 class="logo-font">conduit</h1>
      <p>A place to share your knowledge.</p>
    </div>
  </div>
</div>
```

任務02：現有模板切版，及元件規劃

- app.component.html 中加入建立好的元件
 - <app-header></app-header>



```
app.component.html
1 | <app-header> </app-header>
2 |
3 | <div class="home-page">
4 |
5 |   <div class="banner">
6 |     <div class="container">
7 |       <h1 class="logo-font">conduit</h1>
8 |       <p>A place to share your knowledge.</p>
9 |     </div>
10 |   </div>
11 |
12 |   <div class="container page">
13 |     <div class="row">
14 |
15 |       <div class="col-md-9">
16 |         <div class="feed-toggle">
17 |           <ul class="nav nav-pills outline-active">
18 |             <li class="nav-item">
```

補充：如果專案不需要測試檔(*.spec.ts)

- 方法1：
- ng new [project name] --skip-tests
- 方法2：
- ng g c [component name] --spec=false
- 方法3：在 angular.json 中進行統一設定
- projects.[project name].schematics
- 參考程式碼



主軸2：Angular 畫面呈現與應用

任務03：使用內嵌繫結顯示資料

- 說明：
 - 請將圖片中紅色框框內的文字內容(title 與 subtitle)，移動到 `app.component.ts` 中
 - 使用內嵌繫結 (Interpolation) 的方式，顯示在畫面上

conduit

Keyword



conduit

A place to share your knowledge.

將文字內容移到 `app.component.ts` 中

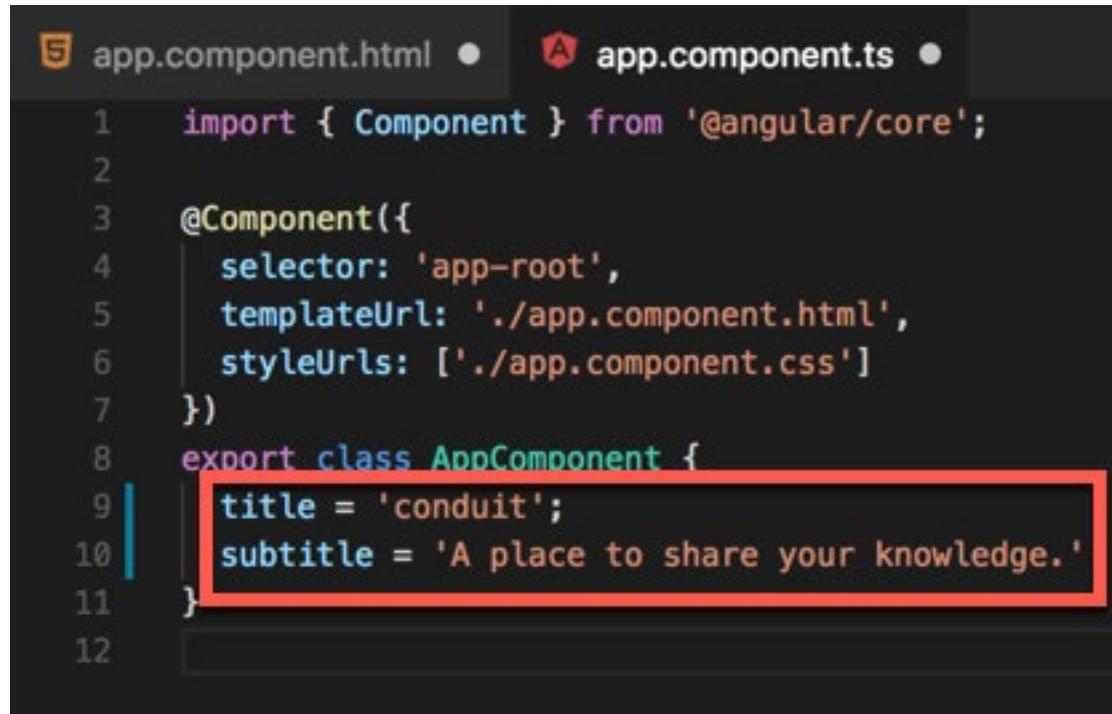
Article List

Popular Tags

programming javascript

任務03：使用內嵌繫結顯示資料

- 在 app.component.ts 中加入 title 與 subtitle 屬性



```
app.component.html ● A app.component.ts ●

1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'conduit';
10  subtitle = 'A place to share your knowledge.'
11 }
12
```

A screenshot of a code editor showing the file app.component.ts. The code defines an Angular component with a selector of 'app-root'. It has a template URL pointing to app.component.html and style URLs pointing to app.component.css. The component is exported. Inside the class definition, there are two properties: 'title' set to 'conduit' and 'subtitle' set to 'A place to share your knowledge.'. The subtitle assignment is highlighted with a red rectangular box.

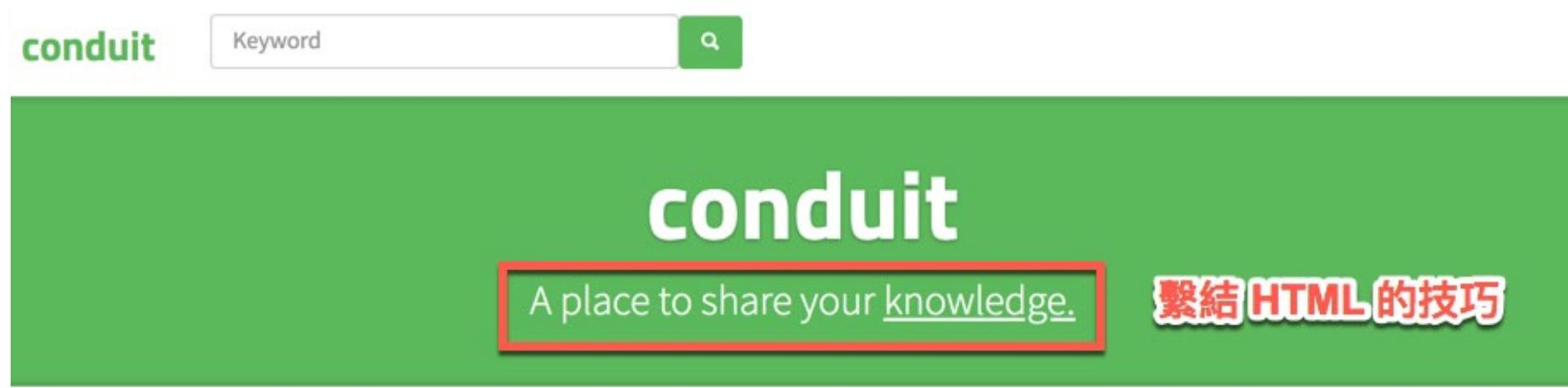
任務03：使用內嵌繫結顯示資料

- 在 app.component.html 中繫結屬性

```
2
3   <div class="home-page">
4
5     <div class="banner">
6       <div class="container">
7         <h1 class="logo-font">{{ title }}
8         <p>{{ subtitle }}</p>
9       </div>
10      </div>
```

任務04：使用屬性繫結顯示資料

- 目標：
 - 改使用屬性繫結將資料呈現在畫面上
 - 請嘗試呈現包含 HTML 標籤的資料



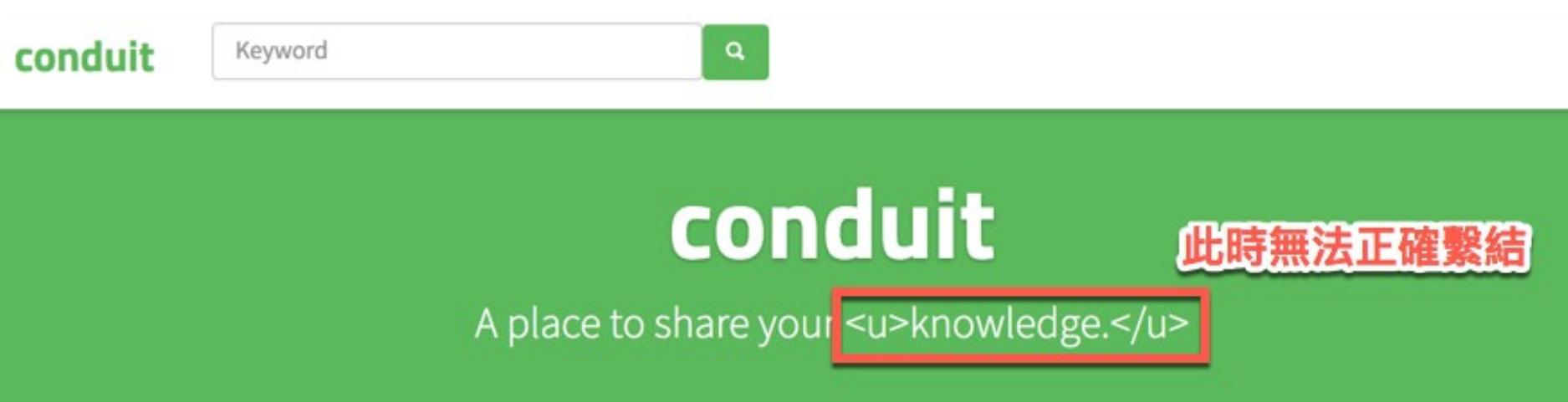
任務04：使用屬性繫結顯示資料

- 使用 `[title]="title"` 繫結資料

```
app.component.html x app.component.ts
1 <app-header></app-header>
2
3 <div class="home-page">
4   <div class="banner">
5     <div class="container">
6       <h1 class="logo-font" [title]="title" {{ title }}></h1>
7       <p>{{ subtitle }}</p>
8     </div>
9   </div>
10 </div>
```

任務04：使用屬性繫結顯示資料

- 將 subtitle 改為包含 HTML 的資料



任務04：使用屬性繫結顯示資料

- 使用屬性繫結：`[innerHTML]="subtitle"`



app.component.html x app.component.ts

```
4
5      <div class="banner">
6          <div class="container">
7              <h1 class="logo-font" [title]="title">{{ title }}</h1>
8              <p [innerHTML]=\"subtitle\"></p>
9          </div>
10     </div>
```

繫結 innerHTML 屬性

補充：如何知道哪些屬性可以繫結

The screenshot shows the Conduit homepage. At the top, there is a search bar with the word "conduit" and a magnifying glass icon. Below the search bar is a green banner with the text "conduit" and "A place to share your knowledge." A small red circle with the number "2" is positioned next to the banner. On the left side, there is a sidebar titled "Article List" featuring a profile picture and name for "Eric Simons" (January 20th) and a "4" in a green box. Below this is an article card for "How to build webapps that scale" by Eric Simons, with a "4" in a green box. To the right of the article is a "Popular Tags" section with tags like "programming", "javascript", "angularjs", "react", "mean", "node", and "rails".

The screenshot shows the browser developer tools open to the "Elements" tab. A red circle with the number "1" is at the top left. The DOM tree shows the structure of the Conduit page. A red circle with the number "2" is on the "Popular Tags" section. A red circle with the number "3" is on the "Properties" tab of the developer tools. A red box highlights the "accessKey" property in the properties panel. A red circle with the number "4" is on the "innerText" property of the "p" element, with the text "找到可以繫結的屬性" (Findable binding properties) overlaid. The properties panel lists various CSS properties and DOM attributes for the selected element.

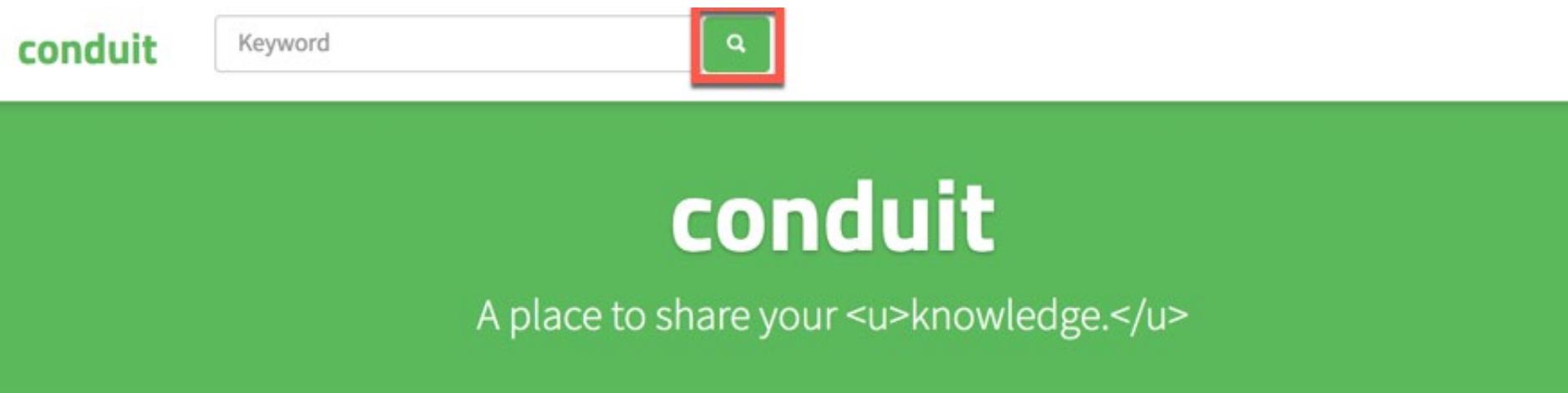
```
<!DOCTYPE html>
<html class="gr__localhost">
  <head>...</head>
  <body data-gr-c-s-loaded="true">
    <app-root _ngcontent-c0 _ngcontent-c1...></app-root>
    <div _ngcontent-c0 class="home-page">
      <div _ngcontent-c0 class="banner">
        <div _ngcontent-c0 class="container">
          <h1 _ngcontent-c0 class="logo-font" title="conduit">conduit</h1>
        ...<p _ngcontent-c0...>A place to share your knowledge.</p>
      </div>
    </div>
  </body>
</html>
```

Properties:

- accessKey: ""
- align: ""
- assignedSlot: null
- attributeStyleMap: StylePropertyMap {size: 0}
- attributes: NamedNodeMap {0: _ngcontent-c0, _ngcontent-autocapitalize: ""}
- baseURI: "http://localhost:4200/"
- childElementCount: 1
- childNodes: NodeList(2) [text, u]
- children: HTMLCollection [u]
- classList: DOMTokenList [value: ""]
- className: ""
- clientHeight: 36
- clientLeft: 0
- clientTop: 0
- clientWidth: 910
- contentEditable: "inherit"
- dataset: DOMStringMap {}
- dir: ""
- draggable: false
- firstChild: text
- firstElementChild: u
- hidden: false
- id: ""
- innerHTML: "A place to share your <u>knowledge.</u>"
- innerText: "A place to share your knowledge."
- inputMode: ""
- isConnected: true
- isContentEditable: false
- lang: ""

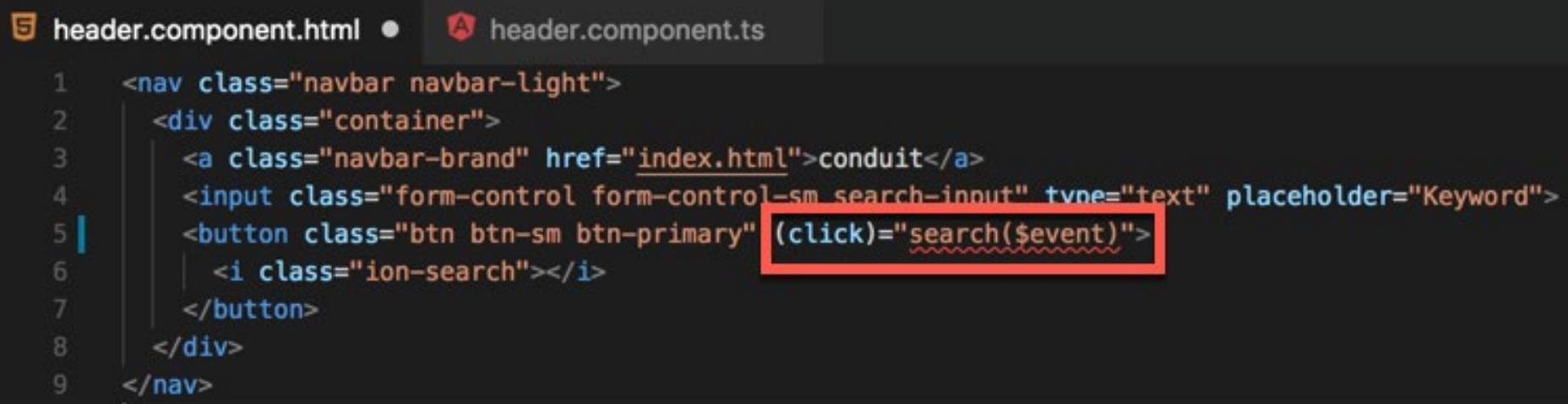
任務05：使用事件繫結

- 目標：
 - 使用事件繫結來得知按鈕按下的事件
 - 請自行嘗試在按下按鈕後改變元件內的一些行為



任務05：使用事件繫結

- header.component.html 中 button 繫結 click 事件
 - `(click)="search($event)"`
 - 提醒：`$event` 是樣板上的固定用法

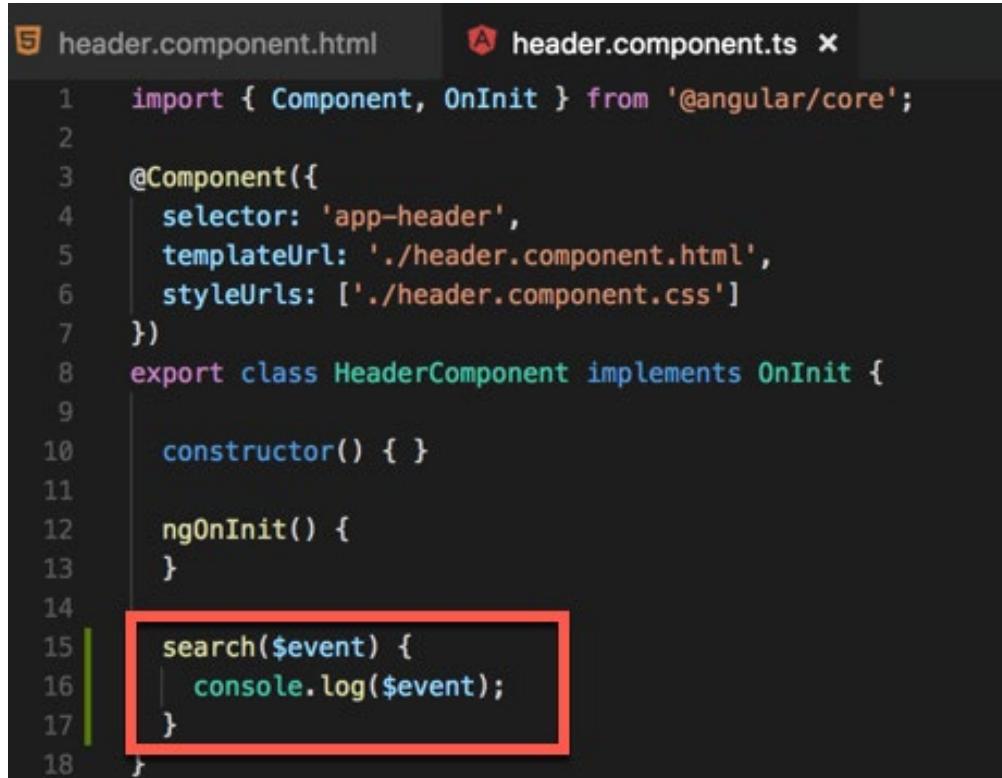


```
header.component.html • header.component.ts
```

```
1 <nav class="navbar navbar-light">
2   <div class="container">
3     <a class="navbar-brand" href="index.html">conduit</a>
4     <input class="form-control form-control-sm search-input" type="text" placeholder="Keyword">
5     <button class="btn btn-sm btn-primary" (click)="search($event)">
6       <i class="ion-search"></i>
7     </button>
8   </div>
9 </nav>
```

任務05：使用事件繫結

- header.component.ts 加上 search 方法



```
header.component.html
header.component.ts ×

1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-header',
5   templateUrl: './header.component.html',
6   styleUrls: ['./header.component.css']
7 })
8 export class HeaderComponent implements OnInit {
9
10   constructor() { }
11
12   ngOnInit() {
13   }
14
15   search($event) {
16     console.log($event);
17   }
18 }
```

任務06：使用雙向繫結

- 目標：
 - 在輸入框使用雙向繫結(two way binding)綁定資料
 - 將綁定的資料顯示在其他地方
 - 透過內嵌繫結或屬性繫結



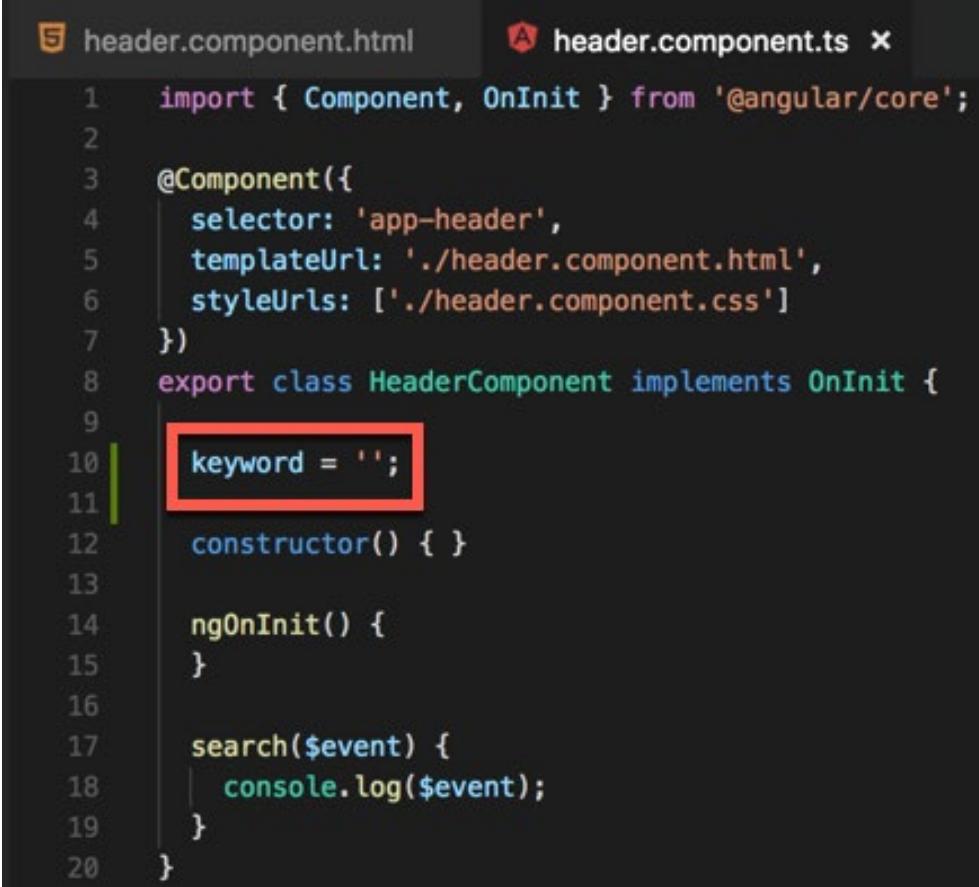
任務06：使用雙向繫結

- 在 app.module.ts 的 imports 中加入 `FormsModule`

```
10 | import { FormsModule } from '@angular/forms';
11 |
12 @NgModule({
13   declarations: [
14     AppComponent,
15     HeaderComponent,
16     FooterComponent,
17     ArticlesComponent,
18     TagsComponent
19   ],
20   imports: [
21     BrowserModule,
22     FormsModule
23   ],
24   providers: [],
25   bootstrap: [AppComponent]
26 })
27 export class AppModule { }
```

任務06：使用雙向繫結

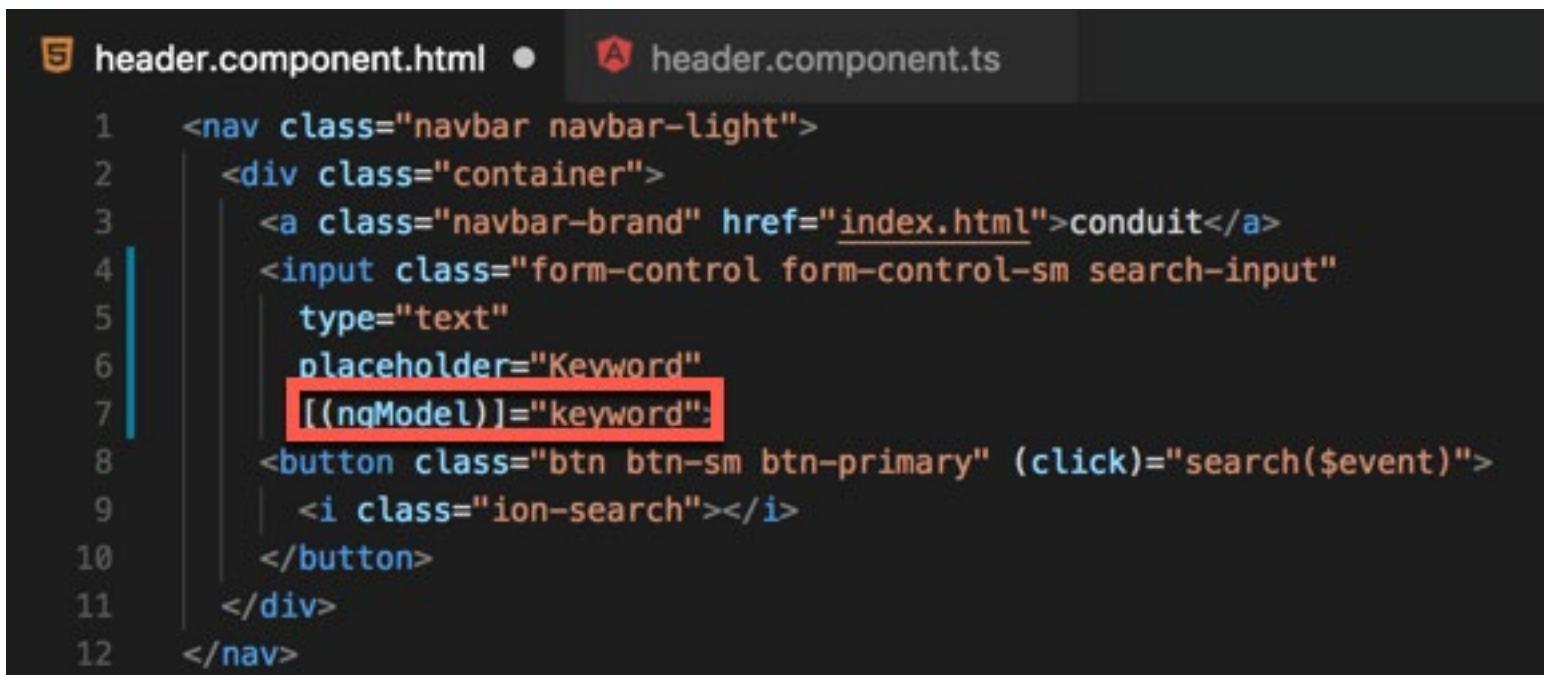
- header.component.ts 中加入 keyword 變數



```
header.component.html header.component.ts ×
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-header',
5   templateUrl: './header.component.html',
6   styleUrls: ['./header.component.css']
7 })
8 export class HeaderComponent implements OnInit {
9
10   keyword = '';
11
12   constructor() { }
13
14   ngOnInit() {
15   }
16
17   search($event) {
18     console.log($event);
19   }
20 }
```

任務06：使用雙向繫結

- header.component.html 的 input 加入雙向繫結
 - [(ngModel)]="keyword"



```
header.component.html • header.component.ts

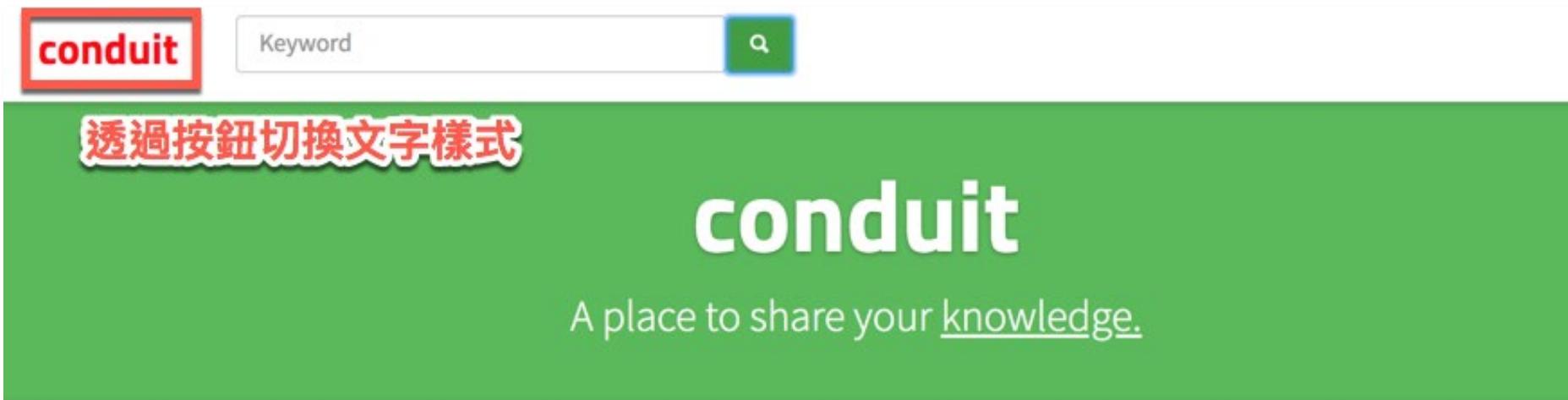
1 <nav class="navbar navbar-light">
2   <div class="container">
3     <a class="navbar-brand" href="index.html">conduit</a>
4     <input class="form-control form-control-sm search-input"
5       type="text"
6       placeholder="Keyword"
7       [(ngModel)]="keyword">
8     <button class="btn btn-sm btn-primary" (click)="search($event)">
9       <i class="ion-search"></i>
10      </button>
11    </div>
12  </nav>
```

補充：雙向繫結的原理

- 雙向繫結其實是 屬性繫結 加上 事件繫結
 - `[ngModel] + (ngModelChange) = [(ngModel)]`

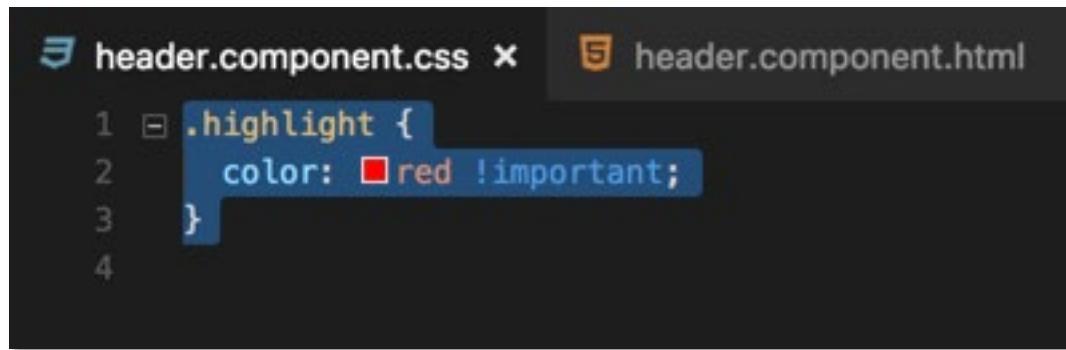
任務07：使用屬性指令(Attribute Directive)

- 目標：
 - 每次按下按鈕時，使用 [class]，替文字切換 highlight 樣式
 - 自行在元件中加入 highlight 樣式
 - 每次按下按鈕時，使用 [style]，放大文字大小



任務07：使用屬性指令(Attribute Directive)

- header.component.css 加入自訂 CSS 樣式



The screenshot shows a code editor with two tabs: 'header.component.css' and 'header.component.html'. The 'header.component.css' tab is active, displaying the following CSS code:

```
1 .highlight {  
2     color: red !important;  
3 }  
4
```

任務07：使用屬性指令(Attribute Directive)

- header.component.ts 加入自訂 highlightTitle 屬性，並在程式中切換狀態

```
export class HeaderComponent implements OnInit {  
  
    keyword = '';  
  
    constructor() { }  
  
    ngOnInit() {  
    }  
  
    highlightTitle = false;  
  
    search($event) {  
        console.log($event);  
  
        this.highlightTitle = !this.highlightTitle;  
    }  
}
```

任務07：使用屬性指令(Attribute Directive)

- header.component.html 中使用 `[class]` 決定是否要顯示 CSS 樣式
 - `[class.highlight] = "highlightTitle"`

任務07：使用屬性指令(Attribute Directive)

- header.component.ts 加入自訂 fontSize 屬性，並在程式中變更資料

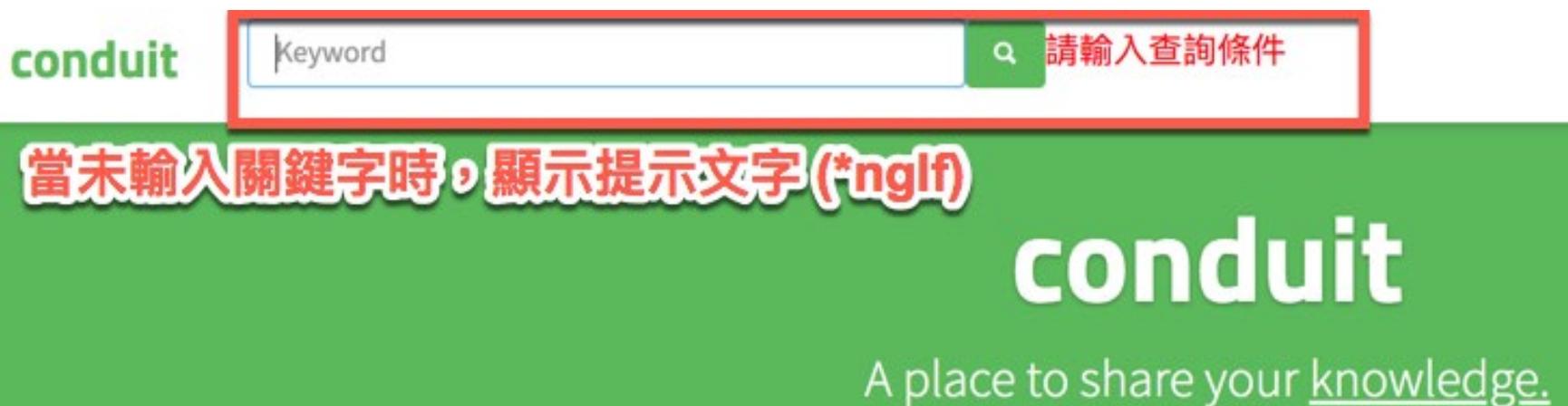
```
export class HeaderComponent implements OnInit {  
  
    keyword = '';  
  
    constructor() { }  
  
    ngOnInit() {  
    }  
  
    highlightTitle = false;  
    fontSize = 24;  
  
    search($event) {  
        console.log($event);  
  
        this.highlightTitle = !this.highlightTitle;  
        ++this.fontSize;  
    }  
}
```

任務07：使用屬性指令(Attribute Directive)

- header.component.html 中使用 **[style]** 設定樣式
 - **[style.fontSize.px] = "fontSize"**

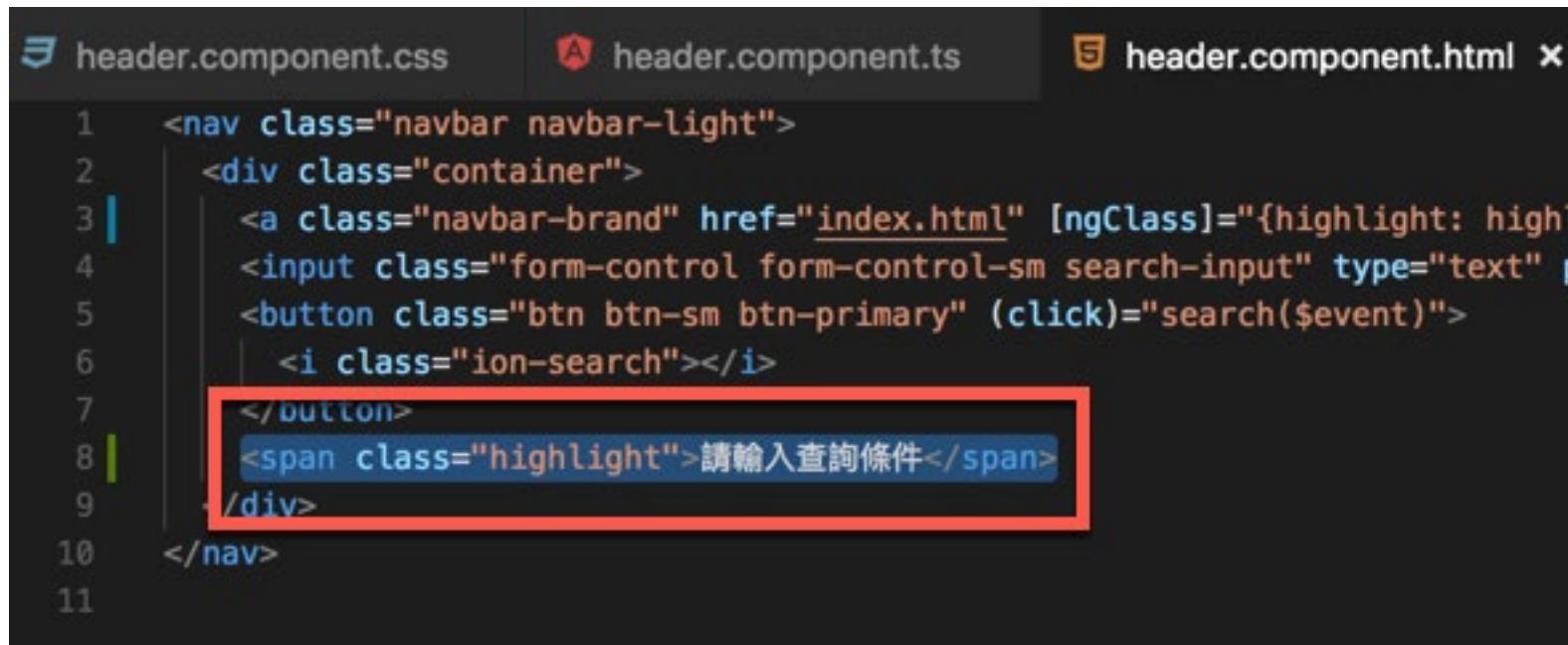
任務08：使用結構指令(Structural Directive)

- 目標 1：當未輸入查詢文字時，使用 *ngIf 提示使用者



任務08：使用結構指令(Structural Directive)

- 在 header.component.html 中加入提示文字



```
header.component.css header.component.ts header.component.html
1 <nav class="navbar navbar-light">
2   <div class="container">
3     <a class="navbar-brand" href="index.html" [ngClass]="{highlight: highlight}">
4       <input class="form-control form-control-sm search-input" type="text" placeholder="請輸入查詢條件" (input)="search($event)">
5     <button class="btn btn-sm btn-primary" (click)="search($event)">
6       <i class="ion-search"></i>
7     </button>
8     <span class="highlight">請輸入查詢條件</span>
9   </div>
10 </nav>
11
```

任務08：使用結構指令(Structural Directive)

- 套用 *ngIf 判斷是否有輸入查詢條件

```
header.component.css header.component.ts header.component.html
```

```
1 <nav class="navbar navbar-light">
2   <div class="container">
3     <a class="navbar-brand" href="index.html" [ngClass]="{highlight: highlight}">Angular 2</a>
4     <input class="form-control form-control-sm search-input" type="text" placeholder="請輸入查詢條件" (input)="search($event)" />
5     <button class="btn btn-sm btn-primary" (click)="search($event)">
6       <i class="ion-search"></i>
7     </button>
8     <span *ngIf="keyword === ''" class="highlight">請輸入查詢條件</span>
9   </div>
10 </nav>
```

補充：*ngIf 搭配 else 應用

- *ngIf 的條件後面可以搭配 else 使用
 - else 後面需要指定一個樣板變數
 - 樣板變數：
 - <ng-template #{{變數名稱}}>內容</ng-template>
 - 樣板變數也是變數，請注意不要與程式內的變數重複

```
</button>
請輸入文字
<ng-template #search>
| 你輸入的文字是：{{ keyword }}
</ng-template>    自訂 search 樣板變數
```

當前面條件為 false 時，
改顯示 search 這個樣板

任務08：使用結構指令(Structural Directive)

- 目標 2：使用 *ngFor 顯示清單資料
 - 資料來源

Article List

sunt aut facere repellat provident occaecati excepturi optio reprehenderit
laudantium enim quasi est quidem magnam voluptate ipsam eos tempora quo necessitatibus dolor quam autem quasi reiciendis et nam sapiente accusantium
[Read more...](#)

quo vero reiciendis velit similique earum
est natus enim nihil est dolore omnis voluptatem numquam et omnis occaecati quod ullam at voluptatem error expedita pariatur nihil sint nostrum voluptatem reiciendis et
[Read more...](#)

odio adipisci rerum aut animi
quia molestiae reprehenderit quasi aspernatur aut expedita occaecati aliquam eveniet laudantium omnis quibusdam delectus saepe quia accusamus maiores nam est cum et ducimus et vero voluptates excepturi deleniti ratione
[Read more...](#)

畫面資料顯示來自 **articles.component.ts** 程式中設定清單資料

任務08：使用結構指令(Structural Directive)

- 在 `articles.components.ts` 中加入文章清單假資料
 - 資料JSON來源

```
import { Component, OnInit, Input } from '@angular/core';

@Component({
  selector: 'app-articles',
  templateUrl: './articles.component.html',
  styleUrls: ['./articles.component.css']
})
export class ArticlesComponent implements OnInit {

  list = [
    {
      "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
      "slug": "zp7yqc",
      "body": "laudantium enim quasi est quidem magnam voluptate ipsam eos\n\ttempora quo necessitatibus\\ndolor quam autem",
      "createdAt": "2018-05-11T21:58:27.358Z",
      "updatedAt": "2018-05-11T21:58:27.358Z",
      "tagList": [],
      "description": "laudantium enim quasi est quidem magnam voluptate ipsam eos\n\ttempora quo necessitatibus\\ndolor quam autem",
      "author": {
        "username": "Eliseo@gardner.biz",
        "bio": "Eliseo",
        "image": "http://placekitten.com/200/300",
        "following": false
      },
      "favorited": false,
      "favoritesCount": 1
    }
  ];
}
```

任務08：使用結構指令(Structural Directive)

- 使用 *ngFor 顯示資料

```
<div class="post-preview" *ngFor="let article of list">
  <div class="post-meta">
    <a href="profile.html">
      <img [src]="article.author.image" />
    </a>
    <div class="info">
      <a href="profile.html" class="author">{{ article.author.username }}</a>
      <span class="date">{{ article.createdAt }}</span>
    </div>
    <button class="btn btn-outline-primary btn-sm pull-xs-right">
      <i class="ion-heart"></i>{{ article.favoritesCount }}
    </button>
  </div>
  <a href="post.html" class="preview-link">
    <h1>{{ article.title }}</h1>
    <p>{{ article.description }}</p>
    <span>Read more...</span>
  </a>
</div>
```

補充：*ngFor的5個變數

- 在 *ngFor 後面可以使用 `let index = index` 的方式取得特定的資料，包含五種變數
 - `index`：資料索引值
 - `first`：是否為第一筆資料
 - `odd`：是否為奇數索引值
 - `even`：是否為偶數索引值
 - `last`：是否為最後一筆資料

```
<div class="post-preview" *ngFor="let article of list; let index = index">
  <div class="post-meta">
    <a href="profile.html">
      <img [src]="article.author.image" />
    </a>
```



主軸3：Angular 中元件與元件傳遞 資料的技巧

任務09：使用 @Input() 傳遞資料給子元件

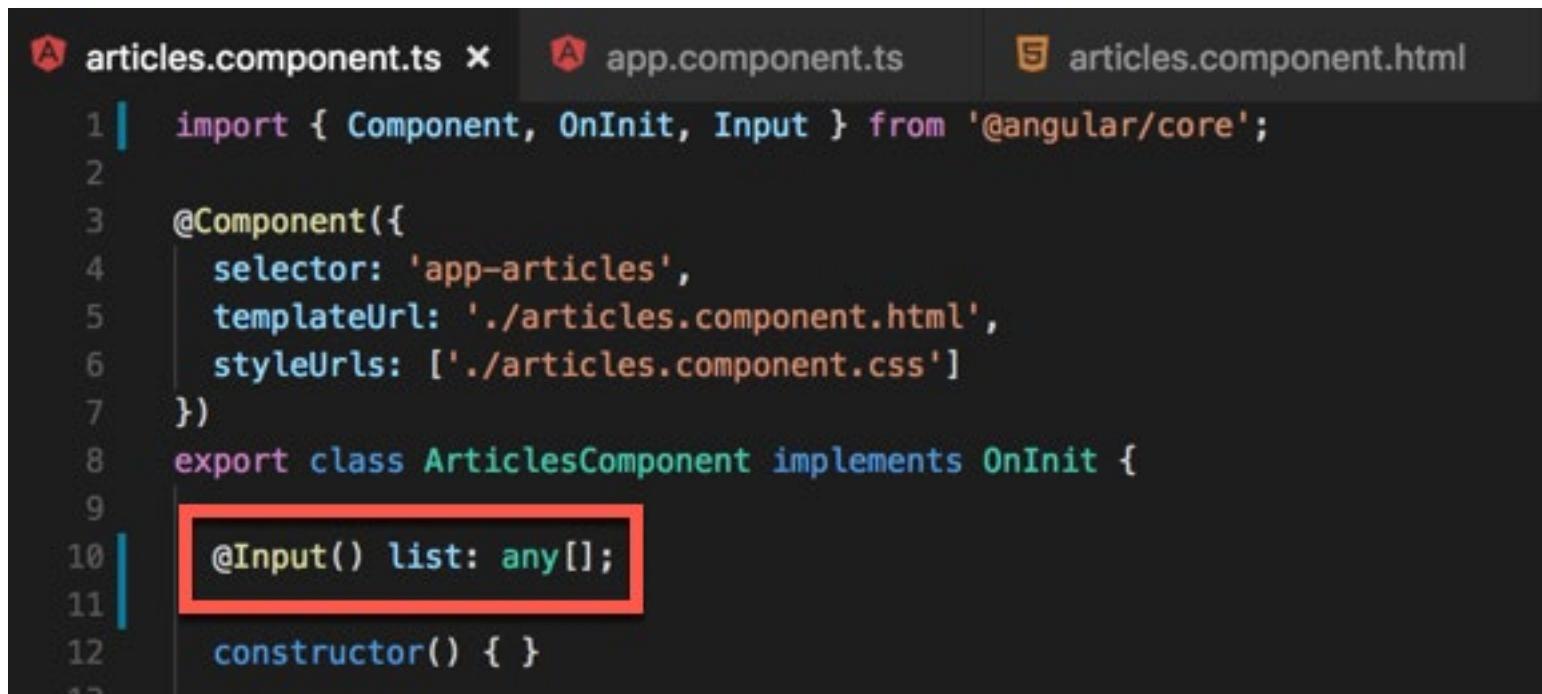
- 目標：
 - 將文章清單資料移動到 AppComponent 中
 - 替 ArticlesComponent 加入 @Input() 接收外部傳來的資料

任務09：使用 @Input() 傳遞資料給子元件

- 將清單資料移到 app.component.ts 中

任務09：使用 @Input() 傳遞資料給子元件

- 將 articles.component.ts 中的list變數改成
 - @Input() list: any[];



```
articles.component.ts
1 import { Component, OnInit, Input } from '@angular/core';
2
3 @Component({
4   selector: 'app-articles',
5   templateUrl: './articles.component.html',
6   styleUrls: ['./articles.component.css']
7 })
8 export class ArticlesComponent implements OnInit {
9
10   @Input() list: any[];
11
12   constructor() { }
```

任務09：使用 @Input() 傳遞資料給子元件

- 將 app.component.html 中的傳入變數給 ArticlesComponent
 - <app-articles [list]="list"></app-articles>

```
articles.component.ts          app.component.ts          app.component.html          articles.component.html
1   <app-header></app-header>
2
3   <div class="home-page">
4
5     <div class="banner">
6       <div class="container">
7         <h1 class="logo-font" [title]="title">{{ title }}</h1>
8         <p [innerHTML]="subtitle"></p>
9       </div>
10      </div>
11
12      <div class="container page">
13        <div class="row">
14
15          <div class="col-md-9">
16            <app-articles [list]="list"></app-articles>
17          </div>
18
19          <div class="col-md-3">
20            <app-tags></app-tags>
21
22
23
```

@Input() 變成元件的屬性

app.component.ts 中的變數名稱

任務10：使用 @Output() 接受元件輸出的資料

- 目標：完成基本搜尋功能
 - 替 HeaderComponent 加入 @Output() 自訂事件
 - 在 AppComponent 中取得 HeaderComponent 中的搜尋結果，並過濾清單資料



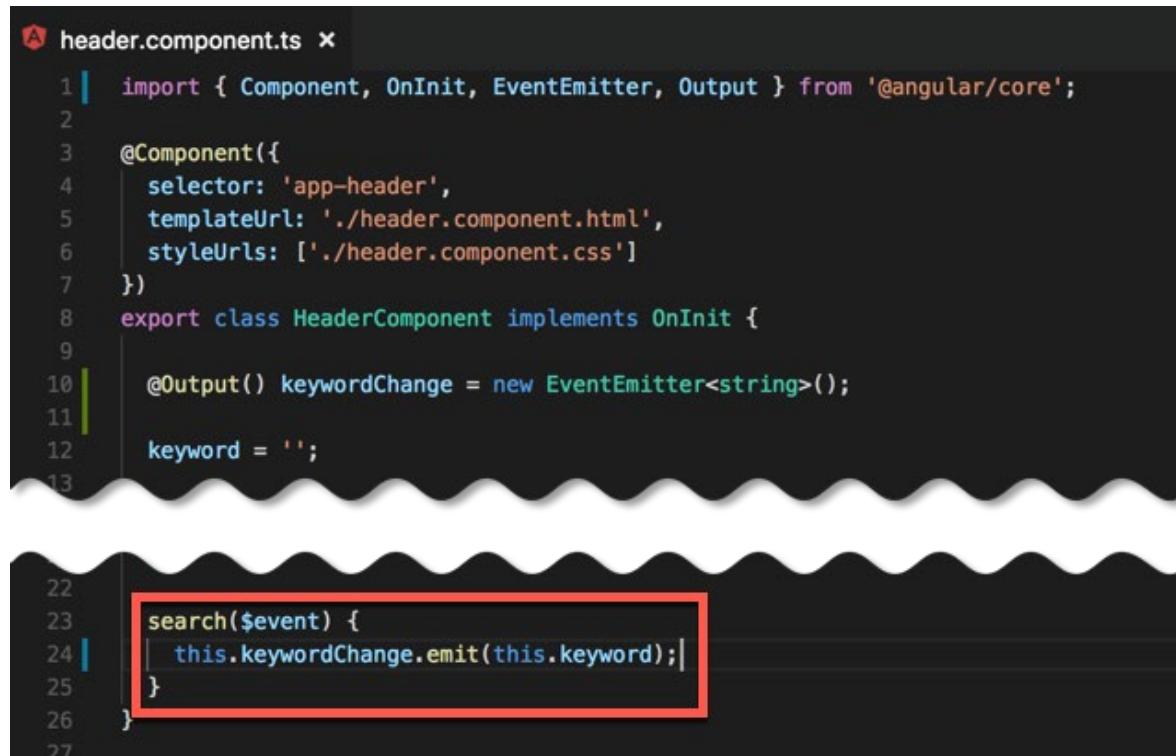
任務10：使用 @Output() 接受元件輸出的資料

- 在 header.component.ts 中加入 @Output() 設定
 - @Output() keywordChange = new EventEmitter<string>();

```
A header.component.ts ✘
1 import { Component, OnInit, EventEmitter, Output } from '@angular/core';
2
3 @Component({
4   selector: 'app-header',
5   templateUrl: './header.component.html',
6   styleUrls: ['./header.component.css']
7 })
8 export class HeaderComponent implements OnInit {
9
10  @Output() keywordChange = new EventEmitter<string>();
11 }
```

任務10：使用 @Output() 接受元件輸出的資料

- 在 header.component.ts 中的 search 方法將查詢條件傳出去
 - `this.keywordChange.emit(this.keyword);`



```
header.component.ts
1 import { Component, OnInit, EventEmitter, Output } from '@angular/core';
2
3 @Component({
4   selector: 'app-header',
5   templateUrl: './header.component.html',
6   styleUrls: ['./header.component.css']
7 })
8 export class HeaderComponent implements OnInit {
9
10   @Output() keywordChange = new EventEmitter<string>();
11
12   keyword = '';
13
14
15
16
17
18
19
20
21
22
23   search($event) {
24     this.keywordChange.emit(this.keyword);
25   }
26
27 }
```

任務10：使用 @Output() 接受元件輸出的資料

- 在 app.component.ts 中實作 searchArticles() 方法

```
export class AppComponent {
  title = 'conduit';
  subtitle = 'A place to share your <u>knowledge.</u>';

  originalList = [...];
  list = this.originalList;

  searchArticles($event) {
    if ($event) {
      this.list = this.originalList.filter(article => article.title.indexOf($event) !== -1);
    } else {
      this.list = this.originalList;
    }
  }
}
```



主軸4：服務 (service) 概念與應用

任務11：自訂服務元件

- 目標：
 - 將文章清單與查詢功能都移動到一個 Service 統一管理
 - 各個元件中不再存放資料與查詢程式，而是統一透過 Service 處理

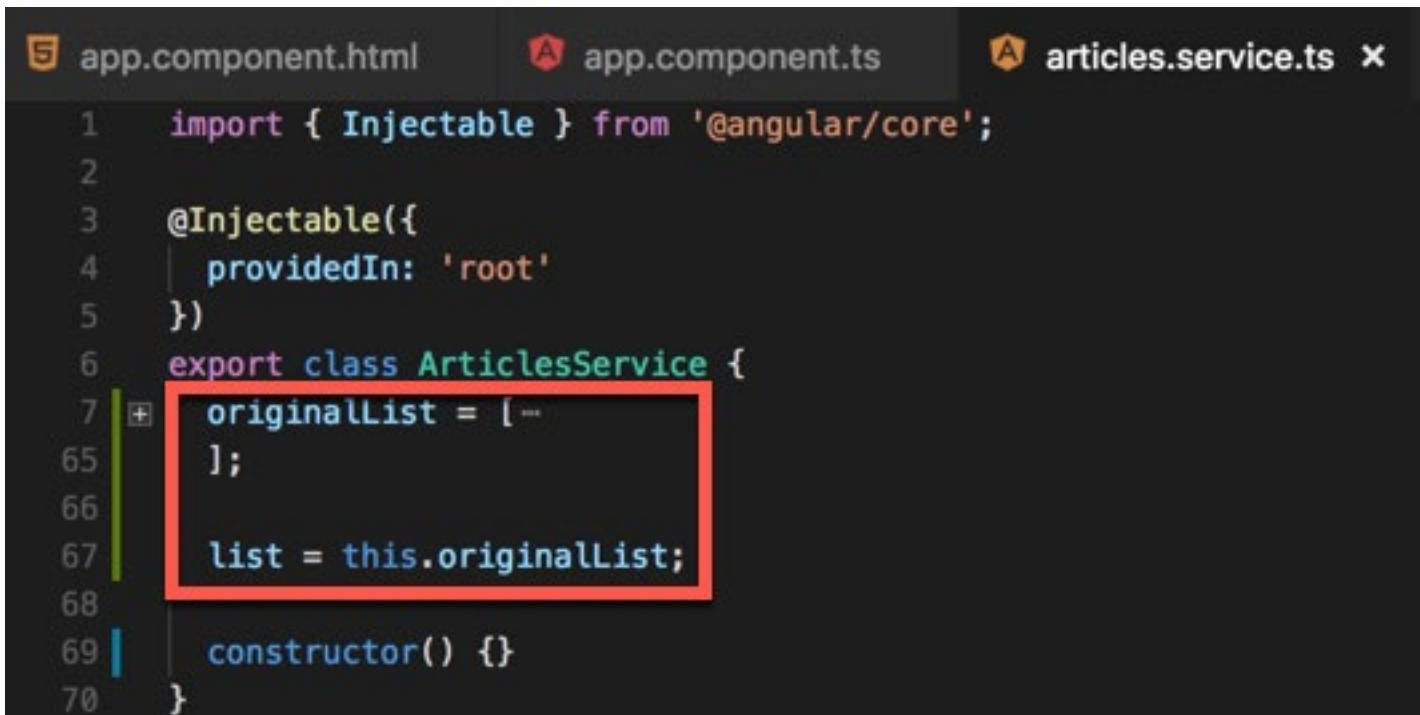
任務11：自訂服務元件

- 建立 ArticlesService
 - ng g s articles

```
wellwind ➤ /Users/wellwind/GitHub/realworld-basic ➤ ↴ mission-10 ✓ ➤ ng g s articles
CREATE src/app/articles.service.ts (137 bytes)
```

任務11：自訂服務元件

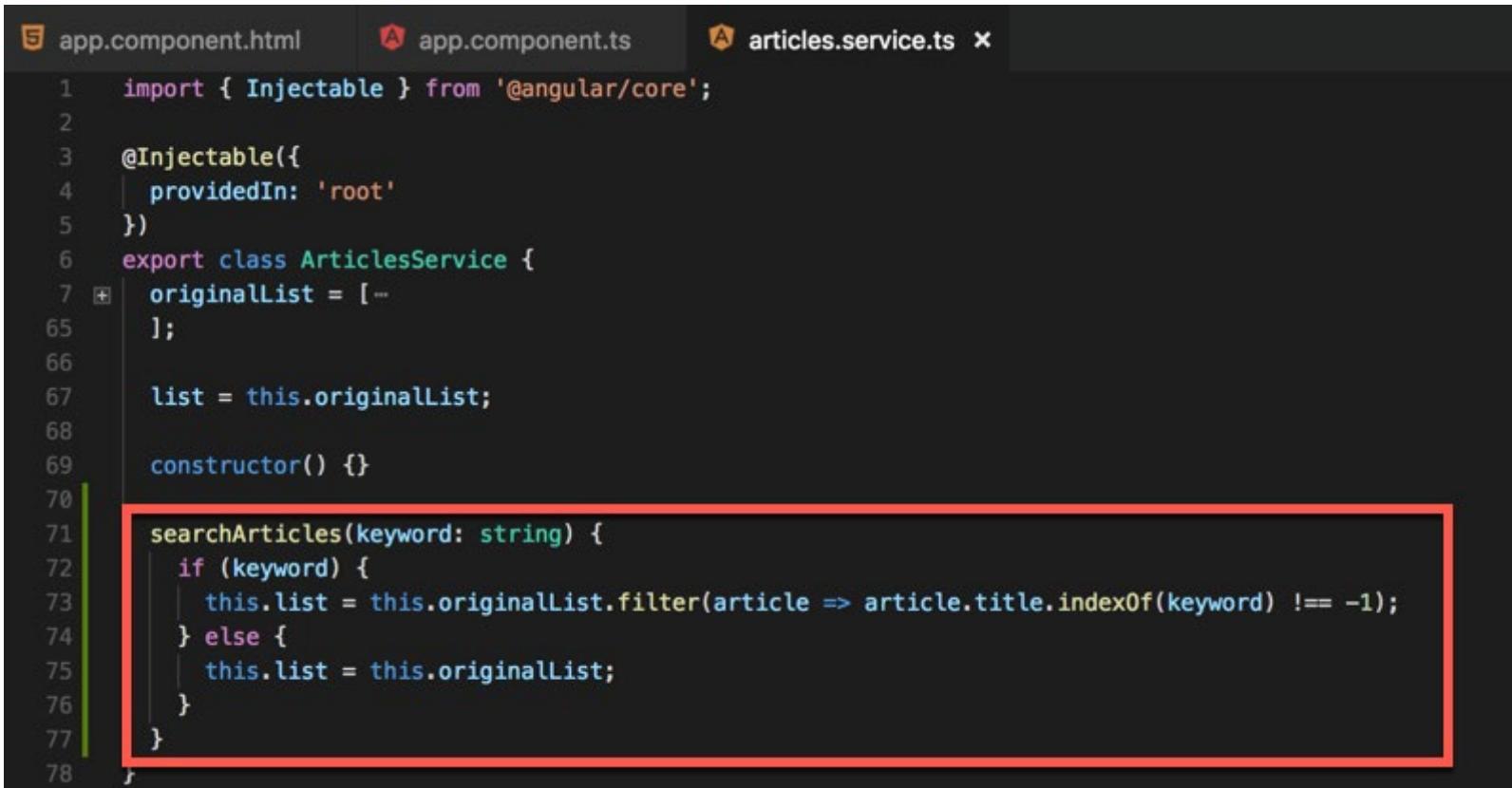
- 將文章相關的資訊移動到 ArticlesService 中



```
app.component.html      app.component.ts      articles.service.ts ×
1   import { Injectable } from '@angular/core';
2
3   @Injectable({
4     providedIn: 'root'
5   })
6   export class ArticlesService {
7     + originalList = [..]
65    ];
66
67    list = this.originalList;
68
69    constructor() {}
70 }
```

任務11：自訂服務元件

- 將搜尋文章的邏輯程式移動到 ArticlesService 中



```
app.component.html app.component.ts articles.service.ts
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class ArticlesService {
7   originalList = [...];
8
9   list = this.originalList;
10
11   constructor() {}
12
13   searchArticles(keyword: string) {
14     if (keyword) {
15       this.list = this.originalList.filter(article => article.title.indexOf(keyword) !== -1);
16     } else {
17       this.list = this.originalList;
18     }
19   }
20 }
```

任務11：自訂服務元件

- 在 app.component.ts 中，注入 ArticlesService

```
export class AppComponent {
  title = 'conduit';
  subtitle = 'A place to share your <u>knowledge.</u>';
  list: any[];
  constructor(private articlesService: ArticlesService) {
    this.list = this.articlesService.list;
  }
}
```

注入 ArticlesService

任務11：自訂服務元件

- 在 app.component.ts 中，使用 ArticlesService 的資料

```
export class AppComponent {  
    title = 'conduit';  
    subtitle = 'A place to share your <u>knowledge.</u>';  
    list: any[];  
  
    constructor(private articlesService: ArticlesService) {  
        this.list = this.articlesService.list;  
    }  
}
```

使用 **ArticlesService** 的資料

任務11：自訂服務元件

- 在 header.component.ts 中，改成使用 ArticlesService 查詢文章

```
header.component.ts ×
1 import { Component, EventEmitter, OnInit, Output } from '@angular/core';
2 import { ArticlesService } from '../articles.service';
3
4 @Component({
5   selector: 'app-header',
6   templateUrl: './header.component.html',
7   styleUrls: ['./header.component.css']
8 })
9 export class HeaderComponent implements OnInit {
10
11   constructor(private articlesService: ArticlesService) {}
12
13   ngOnInit() {}
14
15   search($event) {
16     this.articlesService.searchArticles(this.keyword);
17   }
18
19 }
20
21
22
23
24 }
25
```

注入 ArticlesService

改成透過 ArticlesService 查詢文章

任務11：自訂服務元件

- 實際測試時，沒有效果，為什麼？

```
app.component.ts ×
1 import { Component } from '@angular/core';
2 import { ArticlesService } from './articles.service';
3
4 @Component({
5   selector: 'app-root',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css']
8 })
9 export class AppComponent {
10   title = 'conduit';
11   subtitle = 'A place to share your <u>knowledge.</u>';
12
13   list: any[];
14
15   constructor(private articlesService: ArticlesService) {
16     this.list = this.articlesService.list;
17   }
18 }
articles.service.ts ×
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class ArticlesService {
7   originalList = [...];
8
9   list = this.originalList;
10
11   constructor() {}
12
13   searchArticles(keyword: string) {
14     if (keyword) {
15       this.list = this.originalList.filter(article => article.title.toLowerCase().includes(keyword));
16     } else {
17       this.list = this.originalList;
18     }
19   }
}
```

ArticlesService 的 list 資料更改了

任務11：自訂服務元件

- 實際測試時，沒有效果，為什麼？

The screenshot shows two code files in a code editor:

- app.component.ts**:

```
1 import { Component } from '@angular/core';
2 import { ArticlesService } from './articles.service';
3
4 @Component({
5   selector: 'app-root',
6   templateUrl: './app.component.html',
7   styleUrls: ['./app.component.css']
8 })
9 export class AppComponent {
10   title = 'conduit';
11   subtitle = 'A place to share your <u>knowledge.</u>';
12
13   list: any[];
14
15   constructor(private articlesService: ArticlesService) {
16     this.list = this.articlesService.list;
17   }
18 }
```

A red box highlights the `list: any[];` declaration, and another red box highlights the assignment `this.list = this.articlesService.list;`.
- articles.service.ts**:

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class ArticlesService {
7   originalList = [...];
8
9   constructor() {}
10
11   searchArticles(keyword: string) {
12     if (keyword) {
13       this.list = this.originalList.filter(article => article);
14     } else {
15       this.list = this.originalList;
16     }
17 }
```

A red box highlights the entire `searchArticles` method.

Annotations in red text are overlaid on the code:

- "但是 AppComponent 的 list 資料沒有跟上" (But the AppComponent's list data did not follow)
- "ArticlesService 的 list 資料更改了" (The ArticlesService's list data has been modified)

任務11：自訂服務元件

- 修正方法1：讓 `ArticlesService` 變成 `public`

The image shows a screenshot of the Visual Studio Code editor with two files open:

- app.component.ts**:
A TypeScript file containing the code for the `AppComponent`. It includes imports for `Component` and `ArticlesService`. The `AppComponent` class has a selector of `'app-root'`, a template URL of `'./app.component.html'`, and style URLs of `'./app.component.css'`. The constructor takes a `public articlesService: ArticlesService` parameter. A red arrow points to this line with the annotation **1. 改為 public**.
- app.component.html**:
An HTML file defining the `<app-root>` component. It contains a header, a banner section with a logo and subtitle, and a main content area. In the main content area, there is a `<app-articles [list]="articlesService.list"></app-articles>` element. A red arrow points to this line with the annotation **2. 直接使用 Service 的資料**.

任務11：自訂服務元件

- 修正方法2：改用 getter

```
export class AppComponent {  
  title = 'conduit';  
  subtitle = 'A place to share your knowledge';  
  get list() {  
    return this.articlesService.list;  
  }  
  constructor(private articlesService: ArticlesService) {}  
}
```

改用 getter 的方式取得
ArticlesService裡面的 list 資料

維持 private

任務12：使用 HttpClient 服務

- 目標：
 - 使用 HttpClient 服務串接真實的 RESTful API
 - API 規格
 - 測試用的 API Endpoint
 - 自行建立 JSON Server

任務12：使用 HttpClient 服務

- 加入 HttpClientModule

```
A app.module.ts ×
1 import { HttpClientModule } from '@angular/common/http';
2 import { NgModule } from '@angular/core';
3 import { FormsModule } from '@angular/forms';
4 import { BrowserModule } from '@angular/platform-browser';
5 import { AppComponent } from './app.component';
6 import { ArticlesComponent } from './articles/articles.component';
7 import { FooterComponent } from './footer/footer.component';
8 import { HeaderComponent } from './header/header.component';
9 import { TagsComponent } from './tags/tags.component';
10
11 @NgModule({
12   declarations: [AppComponent, HeaderComponent, FooterComponent, ArticlesComponent, TagsComponent],
13   imports: [BrowserModule, FormsModule, HttpClientModule], // 這行被高亮
14   providers: [],
15   bootstrap: [AppComponent]
16 })
17 export class AppModule {}
```

任務12：使用 HttpClient 服務

- 在 ArticleService 注入 HttpClient 並呼叫 API

```
@Injectable({
  providedIn: 'root'
})
export class ArticleService {
  list: any[];
  keyword: string;

  constructor(private httpClient: HttpClient) {}

  loadArticles() {
    this.httpClient.get('https://conduit.productionready.io/api/articles').subscribe((response: any) => {
      this.list = response.articles;
    });
  }

  getArticles(): Observable<any[]> {
    return this.httpClient.get('https://conduit.productionready.io/api/articles')
      .pipe(map((response: any) => response.articles));
  }
}
```

注入 HttpClient

使用 get 方法取得 API 資料

任務12：使用 HttpClient 服務

- 在 AppComponent 初始化時，取得 API 資訊

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  title = 'conduit';
  subtitle = 'A place to share your <u>knowledge.</u>';

  list: any[];

  constructor(private articlesService: ArticlesService) {}

  ngOnInit() {
    this.articlesService
      .getArticles()
      .subscribe(articles => {
        this.list = articles;
      });
  }
}
```

實作 OnInit
生命週期方法

在元件初始化時，
讀取API資料

補充：避免CORS的開發技巧

- 適用 web 與 API 計畫發布在同一台主機上，但開發時無法跨網域存取API的情境
 - 加入 proxy.config.json
 - 設定檔範本
 - `ng serve --proxy-config proxy.config.json`
 - 參考文章



主軸5：其他 Angular 功能特色

任務13：使用Angular內建的Pipe

- 目標：
 - 使用 DatePipe 來改變日期顯示格式
 - 格式：yyyy-MM-dd



The screenshot shows a user profile with a gravatar icon, the name "1234Test1234", and the date "2018-05-15". Below the profile is a button labeled "改變日期顯示格式". The main content area contains the heading "Markdown Test" and the text "Testing markdown syntax". A "Read more..." link is also visible.

Markdown Test

Testing markdown syntax

Read more...

任務13：使用Angular內建的Pipe

- 在 ArticlesComponent 中使用 DatePipe

- {{ article.createdAt | date: 'yyyy-MM-dd' }}

```
<div class="post-preview" *ngFor="let article of list; let articleIndex = index">
  <div class="post-meta">
    <a href="profile.html">
      <img [src]="article.author.image" />
    </a>
    <div class="info">
      <a href="profile.html" class="author">{{ article.author.username }}</a>
      <span class="date">{{ article.createdAt | date: 'yyyy-MM-dd' }}</span>
    </div>
    <button class="btn btn-outline-primary btn-sm pull-xs-right" (click)="onAddToFavorites(article)">
      <i class="ion-heart"></i> {{ favoritesCount }}
    </button>
  </div>
  <a href="post.html" class="preview-link">
    <h1>{{ article.title }}</h1>
    <p>{{ article.description }}</p>
    <span>Read more...</span>
  </a>
</div>
```

Pipe名稱

Pipe參數

補充：好用的 AsyncPipe (1)

- 在 app.component.ts 取得 ArticlesService 的 Observable 物件

```
export class AppComponent implements OnInit {
  title = 'conduit';
  subtitle = 'A news app';
  list$: Observable<any>[];

  constructor(private articlesService: ArticlesService) {}

  ngOnInit() {
    this.list$ = this.articlesService.getArticles();
  }
}
```

Observable物件習慣用\$結尾

取得 Observable 物件

補充：好用的 AsyncPipe (2)

- 在 app.component.html 中，使用 AsyncPipe，幫助我們訂閱 Observable 物件

```
<div class="col-md-9">  
    <app-articles [list]="list$ | async"></app-articles>  
</div>
```

使用 **async pipe** 處理非同步資料

任務14：自訂 Pipe

- 目標：
 - 建立一個自訂的 FilterArticlePipe
 - 將過濾文章的程式邏輯移動到 FilterArticlePipe 中
 - 頁面透過 FilterArticlePipe 來過濾文章

任務14：自訂 Pipe

- 建立 Pipe

- ng g p filter-article

```
wellwind ➔ /Users/wellwind/GitHub/realworld-basic ➔ ⌂ mission-11 ✘ ★ ➔ ng g p filter-article
CREATE src/app/filter-article.pipe.ts (215 bytes)
UPDATE src/app/app.module.ts (820 bytes)
```

任務14：自訂 Pipe

- 在建立的 FilterArticlePipe 中的 transform 方法加入程式



```
  pipe: PipeTransform
})
export class FilterArticlePipe implements PipeTransform {
  transform(articles: any[], keyword?: any): any {
    if (articles && keyword) {
      return articles.filter(article => article.title.indexOf(keyword) !== -1);
    } else {
      return articles;
    }
  }
}
```

任務14：自訂 Pipe

- 在 ArticlesService 中，搜尋文章功能改成紀錄搜尋關鍵字

```
@Injectable({
  providedIn: 'root'
})
export class ArticlesService {
  list: any[];
  keyword: string;

  constructor(private httpClient: HttpClient) {}

  loadArticles() {
    this.httpClient.get('https://conduit.productionready.io/api/articles').subscribe((response) =>
      this.list = response.articles;
    );
  }

  getArticles(): Observable<any[]> {
    return this.httpClient.get('https://conduit.productionready.io/api/articles').pipe(map((response) =>
      this.list = response.articles;
    ));
  }

  searchArticles(keyword: string) {
    this.keyword = keyword;
  }
}
```

任務14：自訂 Pipe

- 在 app.component.ts 中，取得 ArticlesService 紀錄的關鍵字

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  title = 'conduit';
  subtitle = 'A place to share your <u>knowledge.</u>';

  list$: Observable<any[]>;
  get keyword() {
    return this.articlesService.keyword;
}
  constructor(private articlesService: ArticlesService) {}

  ngOnInit() {
    this.list$ = this.articlesService.getArticles();
  }
}
```

任務14：自訂 Pipe

- 在 app.component.html 中，使用自訂的 FilterArticlePipe

```
<div class="col-md-9">  
    <app-articles [list]="list$ | async | filterArticle:keyword"></app-articles>  
</div>
```

多個 Pipe 可以透過“|”一起使用



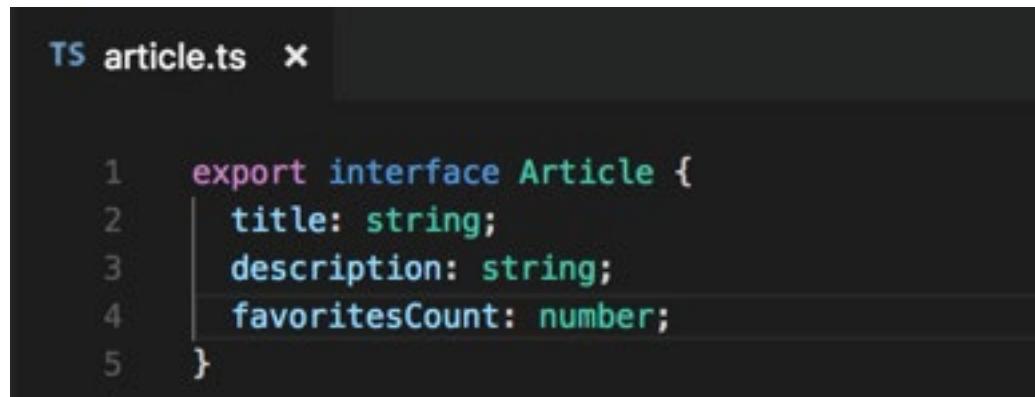
補充內容

TypeScript (1)

- 如何為變數宣告型別
 - 宣告數值型別 : `let num: number;`
 - 宣告字串型別 : `let str: string;`
 - 宣告陣列型別 :
 - `let numbers: number[];`
 - `let numbers: Array<number>;`
 - 自行定義型別 :
 - `interface Article { title: string, likes: string }`
 - `let articles: Article[];`

TypeScript (2)

- 使用 Angular CLI 產生 interface
 - ng g i article
 - 打開 article.ts
 - 定義屬性類型

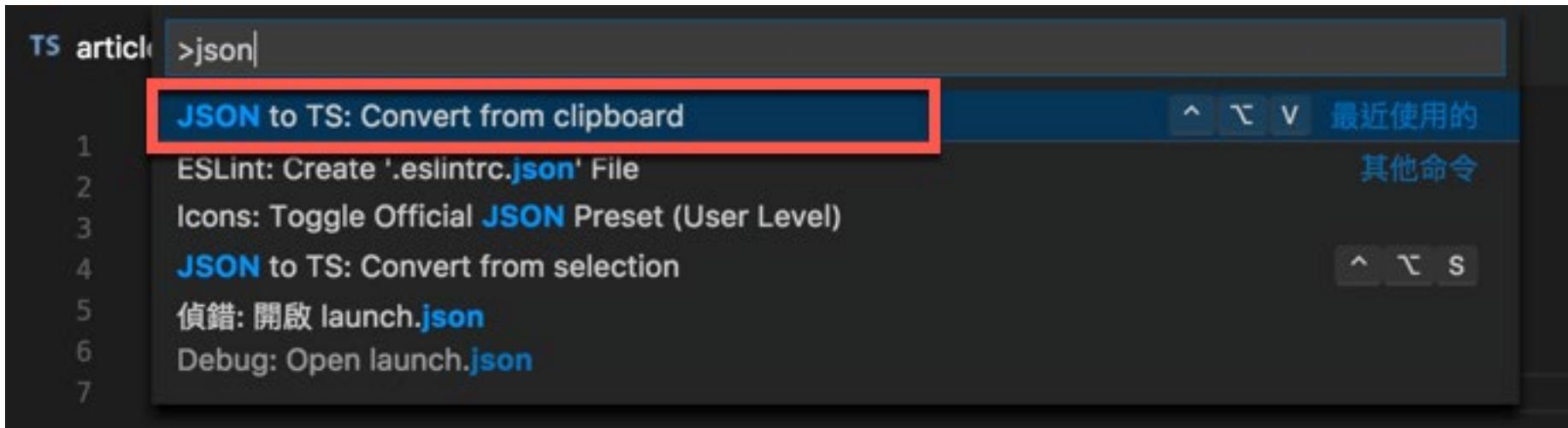


```
TS article.ts ×

1  export interface Article {
2    title: string;
3    description: string;
4    favoritesCount: number;
5 }
```

TypeScript (3)

- 使用 JSON to TS 套件快速建立型別介面定義
 - 安裝 [JSON to TS 套件](#)
 - 複製 JSON 內容
 - VS Code 中使用 Ctrl + Shift + P
 - 輸入 JSON to TS: Convert from clipboard



TypeScript (4)

- 在VS Code 宣告型別後，可以體驗到強型別的好處

The screenshot shows a portion of a TypeScript component definition:

```
@Component({
  selector: 'app-articles',
  templateUrl: './articles.component.html',
  styleUrls: ['./articles.component.css']
})
export class ArticlesComponent implements OnInit {

  @Input() list: Article[]; // This line is highlighted with a red box

  constructor() { }

  ngOnInit() {
    this.list[0]. // This triggers the auto-complete dropdown
  }
}
```

A red box highlights the type annotation `list: Article[];`. A tooltip with the text "因為宣告型別的關係，相關屬性可以 auto complete" is overlaid on the code. An auto-complete dropdown is visible at the bottom right, showing properties of the `Article` type: `description`, `favoritesCount`, and `title`. The `title` property is currently selected.

RxJS (1)

- 在 Angular 中，幾乎所有的非同步行為都是透過 RxJS
- 加入資料流的概念，將資料封裝成一系列的事件
- 使用 `Subscribe()` 來並取得事件內容

RxJS (2)

- Pipeable Operators

- 透過 pipe() 方法，搭配 Operators 來處理資料流

```
getArticles(): Observable<any> {
  return this.httpClient.get('https://conduit.productionready.io/api/articles').pipe(
    map((response: any) => response.articles)
  );
}
```

使用 pipe() 組合多個 operators →

map() 是 RxJS 其中一種 operators，
概念與 JavaScript 的 map() 相似



相關資源整理



相關資源整理

- [Angular 文件\[英文\]](#)
- [Angular 文件\[簡中\]](#)
- [Angular 文件\[繁中\]](#)
- [Angular CLI](#)
- [TypeScript](#)
- [RxJS](#)



多奇·教育訓練

THANK YOU!

Q&A