

Angular 9 開發實戰 新手實作篇

(詳細解答本)



多奇數位創意有限公司

技術總監 黃保翕 (Will 保哥)

部落格：<http://blog.miniasp.com/>



Introduction

前置準備

準備開發環境

- 參考 [Angular 9 開發環境說明](#) 進行安裝設定
 - 安裝 Git 版控工具
 - 安裝 Node.js v12.16.1 LTS 以上版本
 - 確認 npm 為 v6.13.4 以上版本
 - 確認 Angular CLI 為 v9.0.5 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

擴充功能: 市集

Angular Extension Pack 2

Angular Extension Pack 0.5.2 23K ★ 5
Some of the most popular (and some I find v...
Loiane Groner 安裝

Angular Extension Pack 1.3.2 105K ★ 4.5
Popular Visual Studio Code extensions for A...
Will 保哥 3 安裝

Angular 6 Snippets - Typ... 6.0.14 2.6M ★ 5
181 Angular Snippets (TypeScript, Html, Ang...
Mikael Morlund

Angular v5 Snippets 2.16.1 2.8M ★ 5
Angular v5 snippets by John Papa
John Papa

Angular Extension Pack 0.0.4 713 ★ 5
List of extensions for Angular development
Plínio Naves 安裝

Angular Extension Pack 0.0.13 107
An extension pack for Angular developers.
rexebin 安裝

Angular Extension Pack 1.0.0 888

擴充功能: Angular Extension Pack x

Angular Extension Pack doggy8088.angular-extension-pack

Will 保哥 | 105,942 | ★★★★★ | 儲存庫 | 授權

Popular Visual Studio Code extensions for Angular Development

安裝

安裝完成後記得重新啟動 Visual Studio Code

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

Extensions Included

Angular Code Snippets



Introduction

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

coolrare / realworld-basic-template

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

No description, website, or topics provided. [Edit](#)

[Manage topics](#)

5 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find File Clone or download

黃升煌 Mike Merge remote-tracking branch 'origin/master'

assets	Update style.css
index.html	移除不必要的內容

Help people interested in this repository understand your project by adding a README.

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/coolrare/realworld

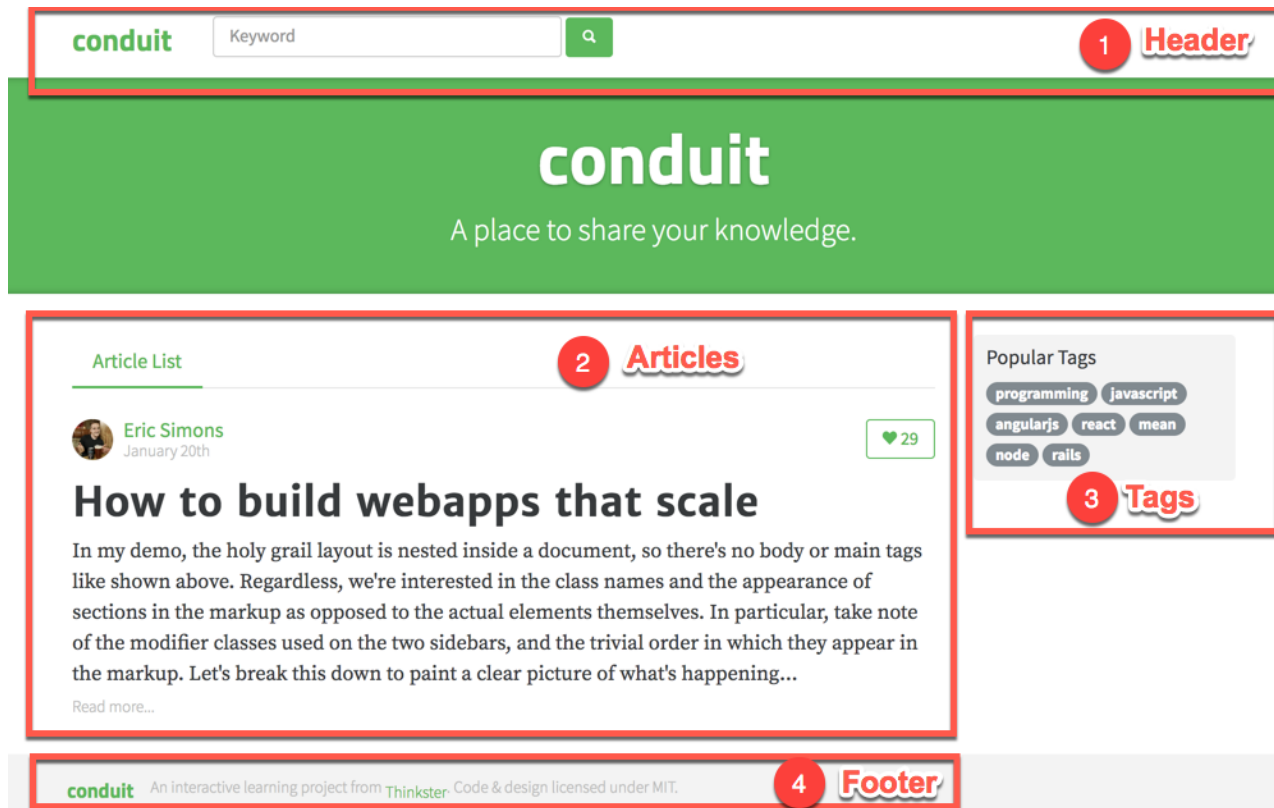
Open in Desktop Download ZIP

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



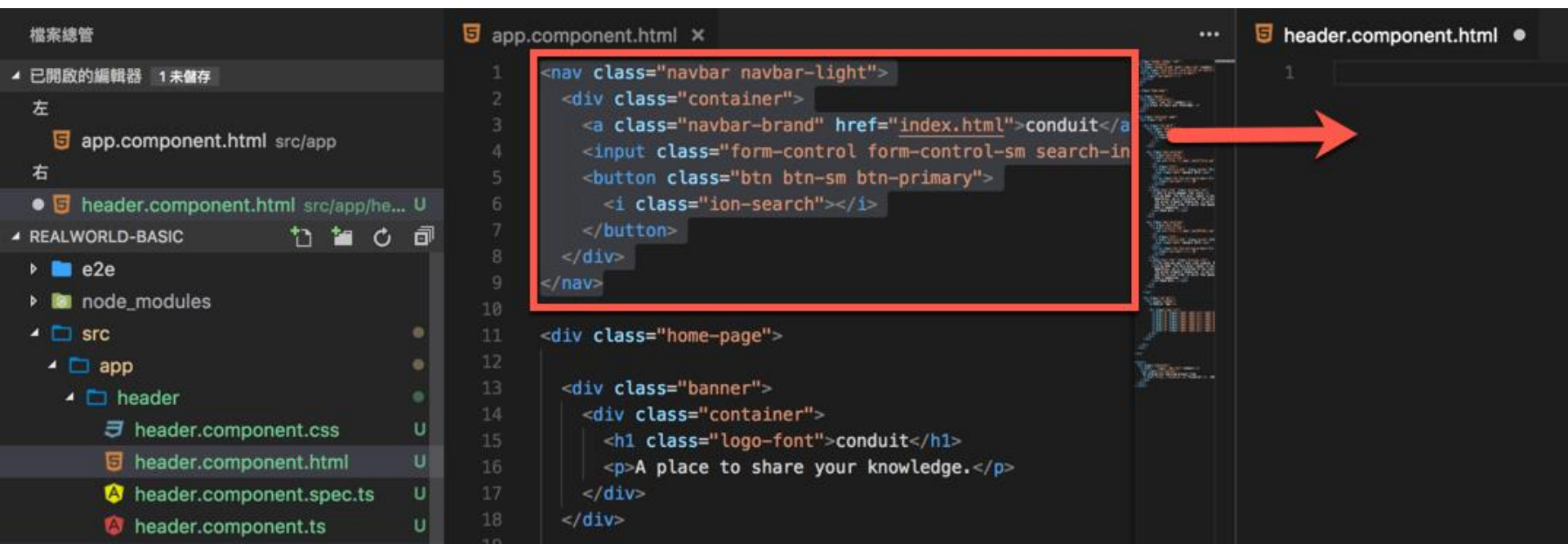
任務02：現有模板切版，及元件規劃

- 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)
```

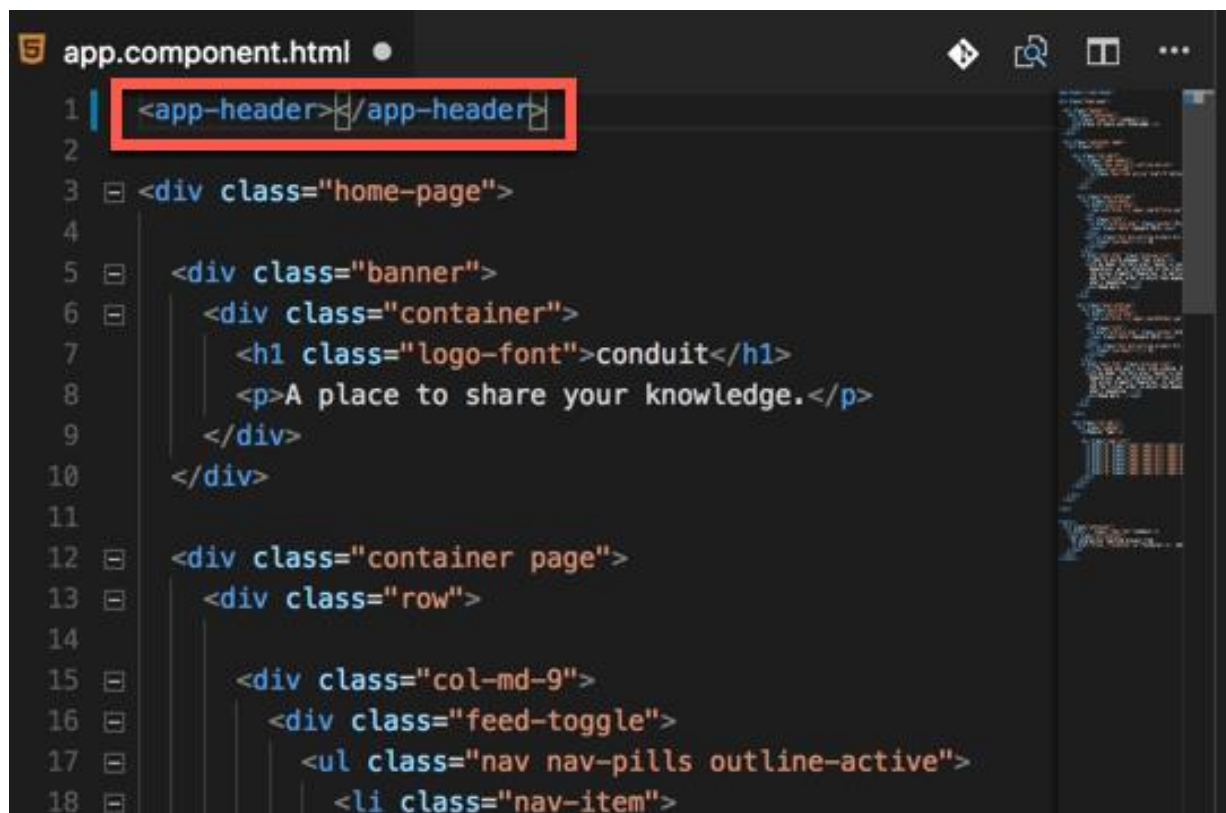
任務02：現有模板切版，及元件規劃

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



任務02：現有模板切版，及元件規劃

- app.component.html 中加入
 - `<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**

補充：如果專案不需要測試檔(*.spec.ts)

- 方法3：在 angular.json 中設定
 - projects.[project name].schematics
 - [程式碼參考](#)

```
"schematics": {
  "@schematics/angular:class": {
    "spec": false
  },
  "@schematics/angular:component": {
    "spec": false
  },
  "@schematics/angular:directive": {
    "spec": false
  },
  "@schematics/angular:guard": {
    "spec": false
  },
  "@schematics/angular:module": {
    "spec": false
  },
  "@schematics/angular:pipe": {
    "spec": false
  },
  "@schematics/angular:service": {
    "spec": false
  }
},
```



Introduction

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



conduit

A place to share your knowledge.

將文字內容移到 `app.component.ts` 中

Article List

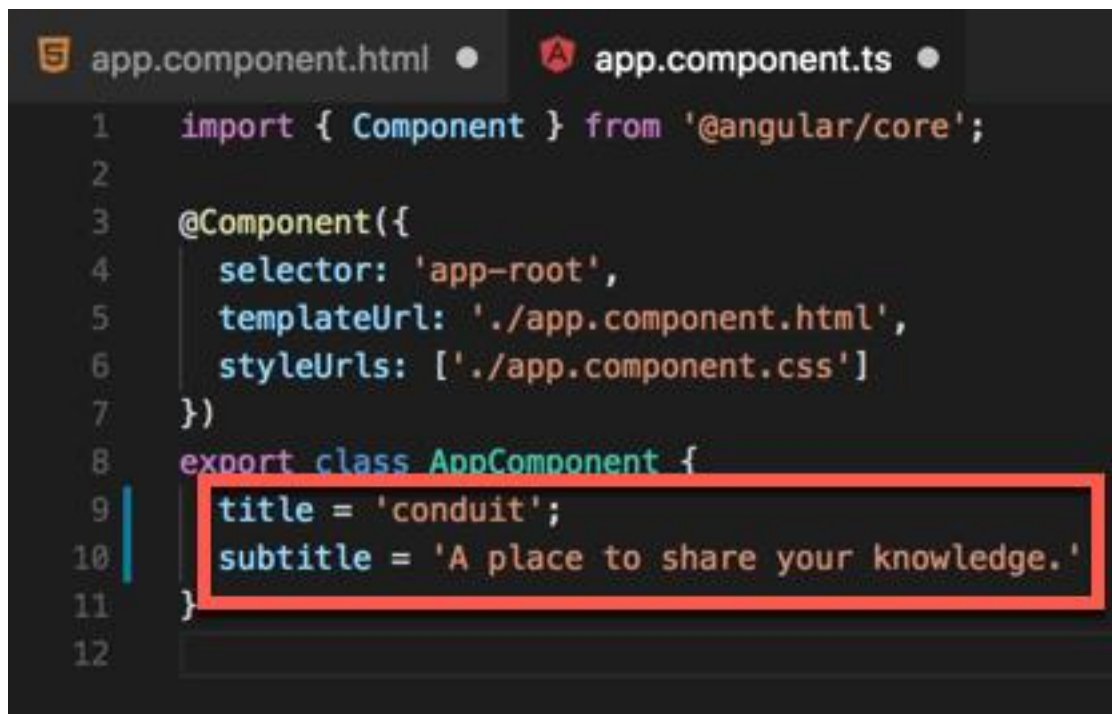
Popular Tags

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 • 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
```

任務03：使用內嵌繫結顯示資料

- 在 `app.component.html` 中繫結屬性

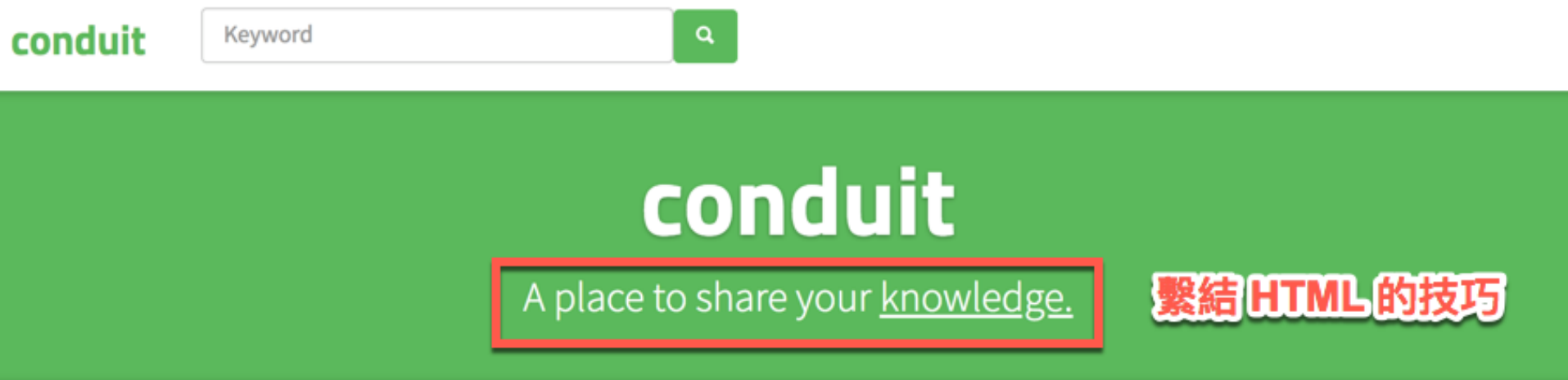


```
2
3 <div class="home-page">
4
5   <div class="banner">
6     <div class="container">
7       <h1 class="logo-font" {{ title }} /h1>
8       <p {{ subtitle }} /p>
9     </div>
10  </div>
```

The screenshot shows a code editor with two tabs: `app.component.html` and `app.component.ts`. The `app.component.html` tab is active, displaying the following HTML code with Angular interpolation syntax. Two red rectangular boxes highlight the binding expressions: one around `{{ title }}` in the `<h1>` tag and another around `{{ subtitle }}` in the `<p>` tag.

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



The screenshot shows a code editor with two tabs: `app.component.html` and `app.component.ts`. The `app.component.html` tab is active, displaying the following HTML template code:

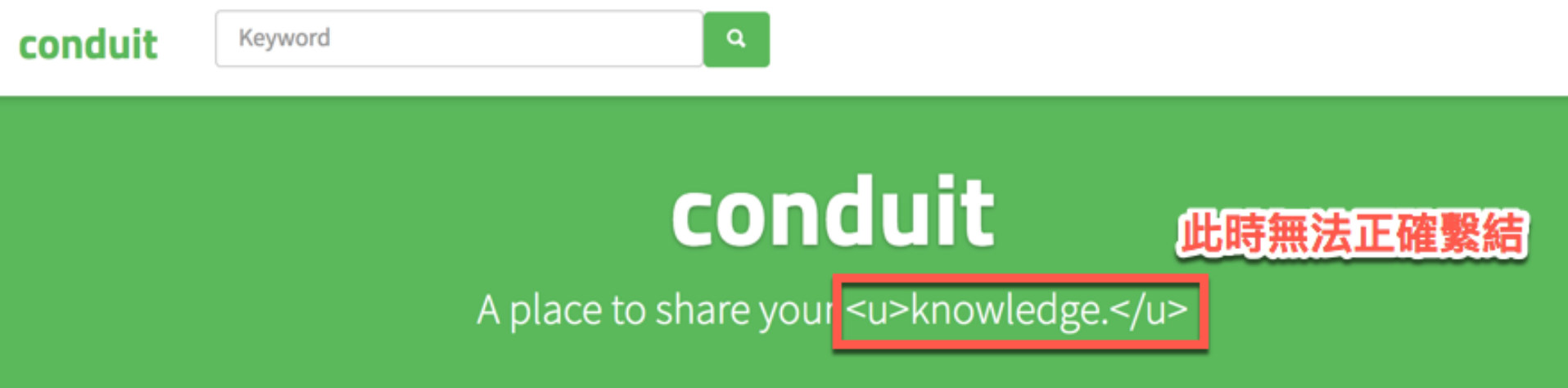
```
1 <app-header></app-header>
2
3 <div class="home-page">
4   <div class="banner">
5     <div class="container">
6       <h1 class="logo-font" [title]="title" {{ title }}</h1>
7       <p>{{ subtitle }}</p>
8     </div>
9   </div>
10 </div>
```

Annotations are present on the code:

- Two red labels, **屬性** (Attribute) and **資料** (Data), are positioned above the `[title]="title"` expression.
- Two red arrows point from these labels to the `[title]="title"` expression, which is enclosed in a red rectangular box.

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"`



The screenshot shows a code editor with two tabs: `app.component.html` and `app.component.ts`. The `app.component.html` tab is active, displaying the following HTML code:

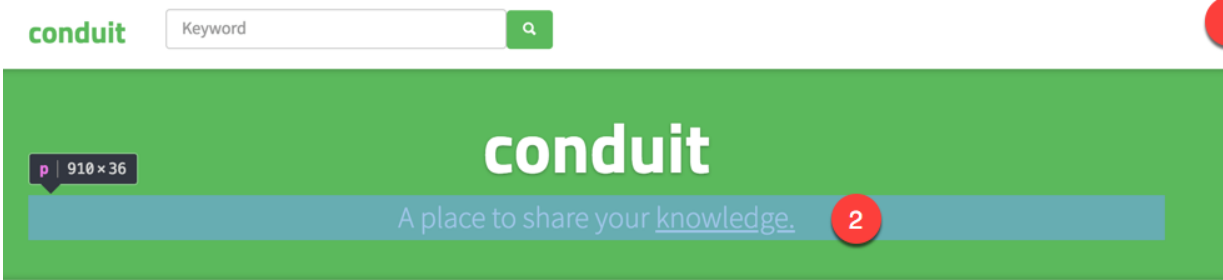
```
4
5 <div class="banner">
6   <div class="container">
7     <h1 class="logo-font" [title]="title" {{ title }}></h1>
8     <p [innerHTML]="subtitle"></p>
9   </div>
10 </div>
```

Annotations on the code:

- A red box highlights the attribute `[innerHTML]="subtitle"` on line 8.
- A red arrow points from the text **繫結 innerHTML 屬性** (Bind innerHTML attribute) to the `[innerHTML]` attribute.

補充：如何知道哪些屬性可以繫結


- Hint :



conduit

A place to share your knowledge.

Article List


 Eric Simons
January 20th

♥ 29

How to build webapps that scale

In my demo, the holy grail layout is nested inside a document, so there's no body or main tags like shown above. Regardless, we're interested in the class names and the appearance of sections in the markup as opposed to the actual elements themselves. In particular, take note of the modifier classes used on the two sidebars, and the trivial order in which they appear in the markup. Let's break this down to paint a clear picture of what's happening...

Read more...

 Albert Pai
January 20th

♥ 32

The song you won't ever stop singing. No matter how hard you try.

In my demo, the holy grail layout is nested inside a document, so there's no body or main tags like shown above. Regardless, we're interested in the class names and the appearance of

Popular Tags

programming javascript
angularjs react mean
node rails

1

Elements Console Sources >>

```
<!DOCTYPE html>
<html class="gr_localhost">
  <head>...</head>
  <body data-gr-c-s-loaded="true">
    <app-root _ngghost-c0 ng-version="6.0.0">
      <app-header _ngcontent-c0 _ngghost-c1>...</app-header>
      <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>...</p>
          </div>
        </div>
      </div>
    </app-root>
  </body>
</html>
```

2

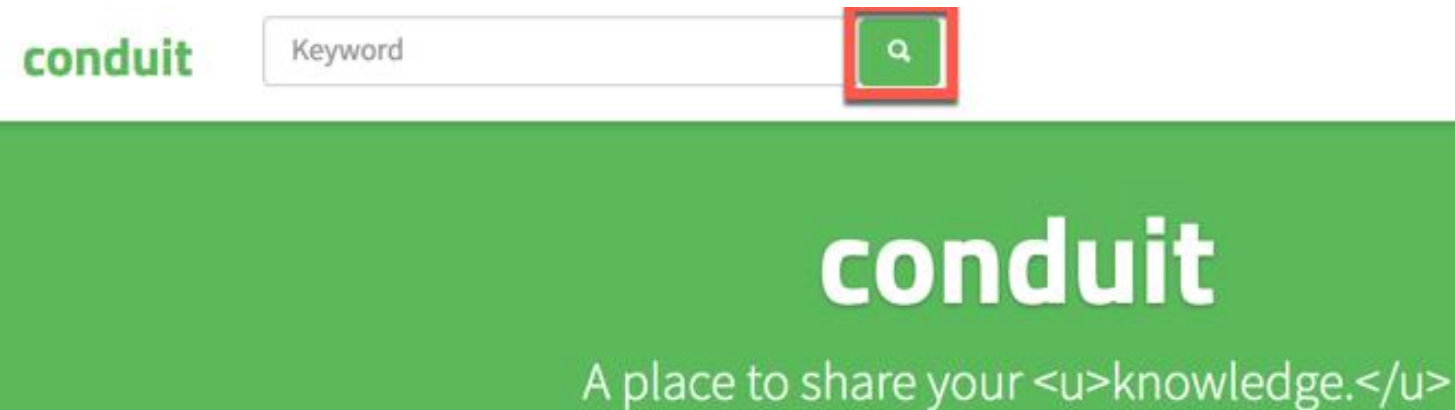
3

4

找到可以繫結的屬性

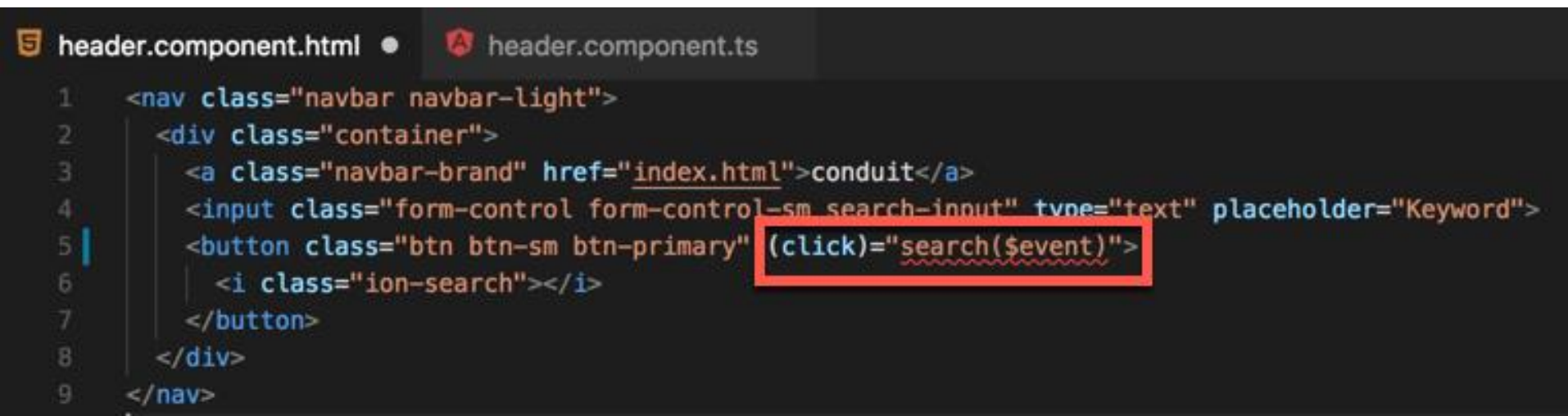
任務05：使用事件繫結

- 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

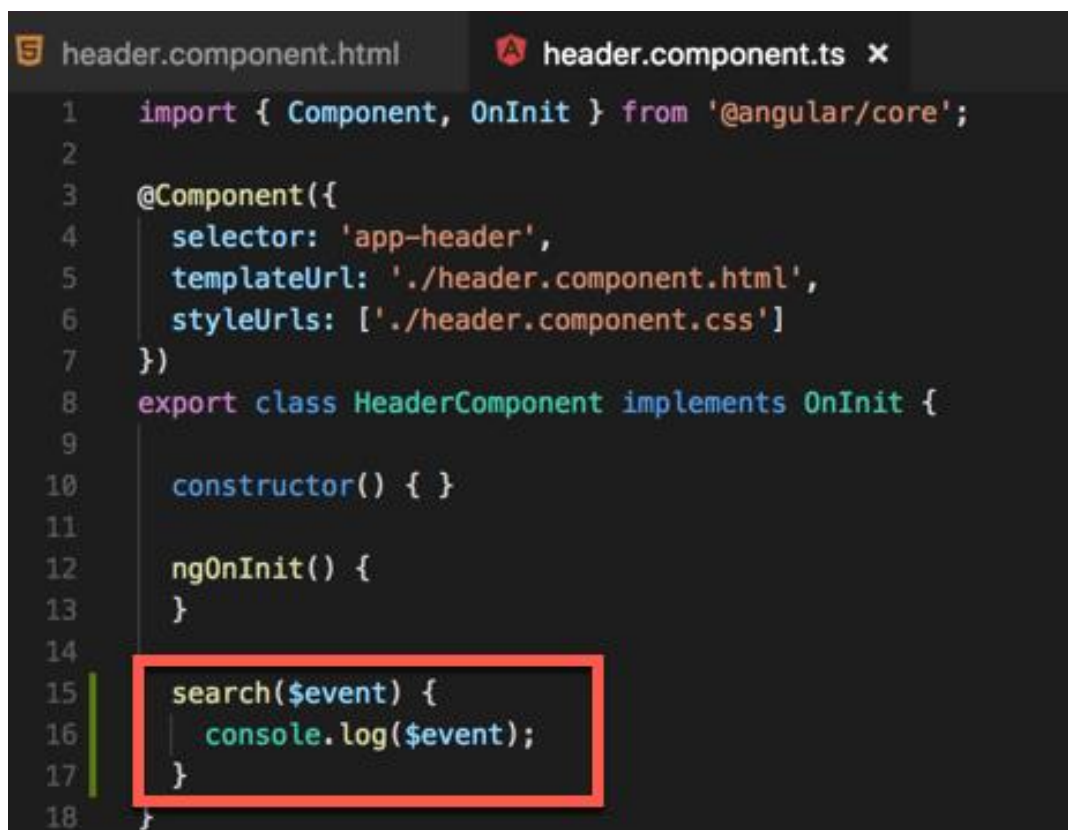
- header.component.html 中 button 繫結 click 事件
 - (click)="search(\$event)"
 - Hint: \$event 是樣板上的固定用法



```
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

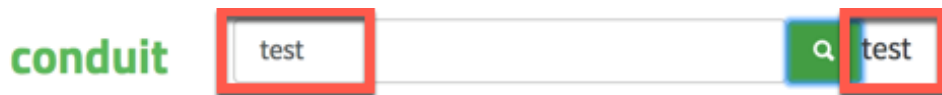


```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-header',
5    templateUrl: './header.component.html',
6    styleUrls: ['./header.component.css']
7  })
8  export class HeaderComponent implements OnInit {
9
10     constructor() { }
11
12     ngOnInit() {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

The image shows a code editor with two tabs: 'header.component.html' and 'header.component.ts'. The 'header.component.ts' tab is active, displaying TypeScript code for an Angular component. The code includes imports for 'Component' and 'OnInit' from '@angular/core', a component decorator with metadata, and a class 'HeaderComponent' that implements 'OnInit'. A new method 'search(\$event)' is added to the class, which logs the event to the console. This new method is highlighted with a red rectangular box.

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



conduit

A place to share your knowledge.

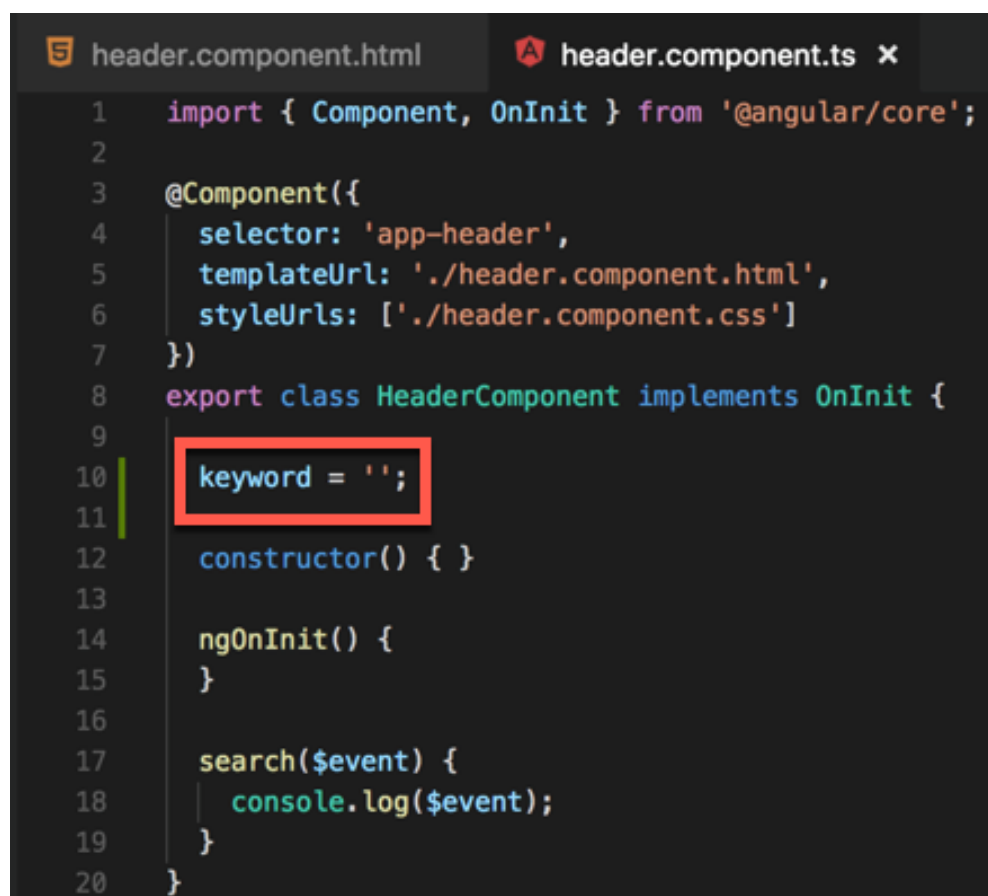
任務06：使用雙向繫結

- 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>
```

任務06：使用雙向繫結

- header.component.ts 中加入 keyword 變數



```
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

- 此時畫面會一片空白，打開 F12 看到錯誤訊息

```
✖ ▶ Uncaught Error: Template parse      compiler.js:215  
errors:  
Can't bind to 'ngModel' since it isn't a known  
property of 'input'. ("</a>  
    <input class="form-control form-control-sm  
search-input" type="text" placeholder="Keyword"  
[ERROR ->][[(ngModel)]]="keyword">  
    <button class="btn btn-sm btn-primary"  
(click)="search($event)">  
    <i ">  
ng:///AppModule/HeaderComponent.html@3:95  
    at syntaxError (compiler.js:215)  
    at
```

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 { }
```

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
– $[\text{ngModel}] + (\text{ngModelChange}) = [(\text{ngModel})]$

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

conduit

Keyword



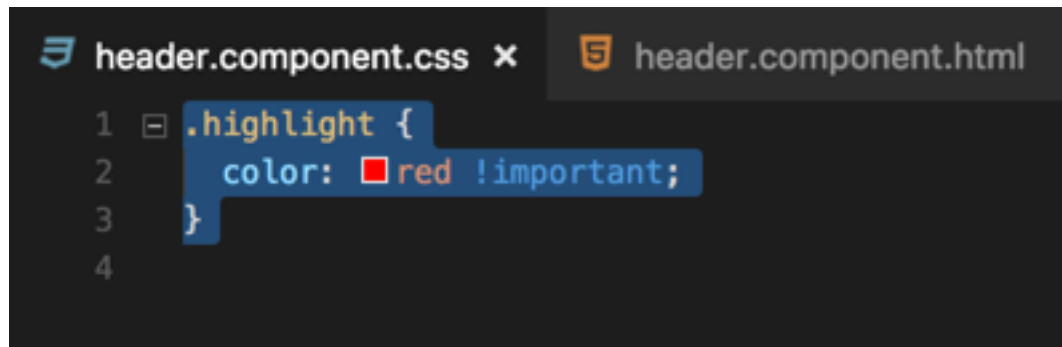
透過按鈕切換文字樣式

conduit

A place to share your knowledge.

任務07：使用屬性指令(Attribute Directive)

- header.component.css 加入自訂 CSS 樣式



A screenshot of a code editor with two tabs: 'header.component.css' and 'header.component.html'. The 'header.component.css' tab is active, showing a CSS rule for '.highlight' with a red color and !important flag. The code is as follows:

```
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，並在程式中允許切換

```
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，並在程式中允許設定

```
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]="{'font-size': fontSize + 'px'}"`

補充：使用 `[class.xxx]` 或 `[style.xxx]`

- 也可以使用 `[class.xxx]` 或 `[style.xxx]` 簡化物件傳入
- ex:
 - `[class.highlight]="highlightTitle"`
 - `[style.font-size.px]="fontSize"`
- 當需要設定的屬性少時非常實用

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

conduit

當未輸入關鍵字時，顯示提示文字 (*ngIf)

conduit

A place to share your knowledge.

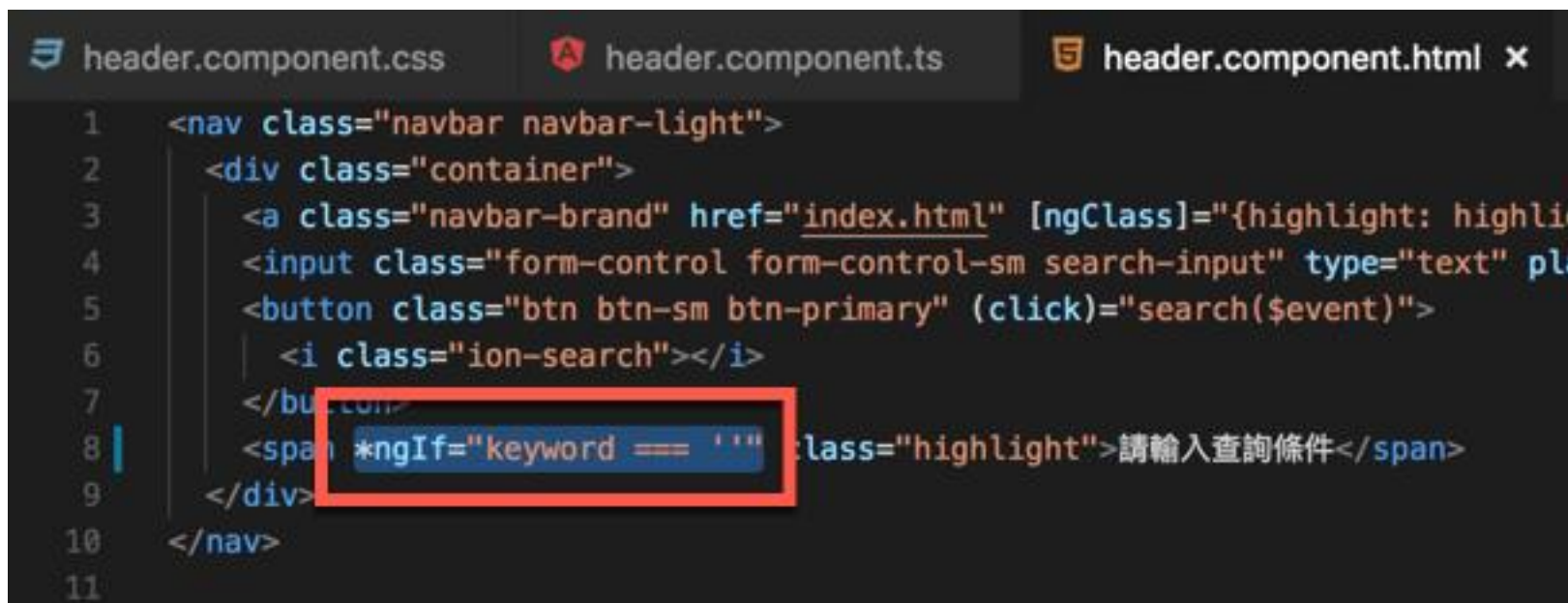
任務08：使用結構指令(Structural Directive)

- 在 header.component.html 中加入提示文字

```
header.component.css  header.component.ts  header.component.html x
1  <nav class="navbar navbar-light">
2    <div class="container">
3      <a class="navbar-brand" href="index.html" [ngClass]="{highlight: highl
4      <input class="form-control form-control-sm search-input" type="text" p
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 x
1  <nav class="navbar navbar-light">
2    <div class="container">
3      <a class="navbar-brand" href="index.html" [ngClass]="{highlight: highli
4      <input class="form-control form-control-sm search-input" type="text" pl
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>
11
```

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>
<span *ngIf="keyword === ''; else search" class="highlight">請輸入文字</span>
<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



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\ntempora 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\ntempora quo necessitatibus\ndolor quam",
      "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 articleIndex = index` 的方式取得特定的資料
 - `index`
 - `first`
 - `odd`
 - `even`
 - `last`

```
<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>  
  </div>  
</div>
```



Introduction

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

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

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

  list = [
    {
      "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
      "slug": "zp7yqc",
      "body": "laudantium enim quasi est quidem magnam voluptate ipsam eos\ntempora quo nulla reprehenderit",
      "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\ntempora quo nulla reprehenderit",
      "author": {
        "username": "Eliseo@gardner.biz",
        "bio": "Eliseo",
        "image": "http://placekitten.com/200/300",
        "following": false
      },
      "favorited": false,
      "favoritesCount": 1
    },
  ],
}
```

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 x  app.component.ts  articles.component.html
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() { }
13 }
```

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>`

```
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
17        <app-articles [list]="list"></app-articles>
18
19      </div>
20
21      <div class="col-md-3">
22
23        <app-tags></app-tags>
```

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>();

```
header.component.ts x
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 | }
```

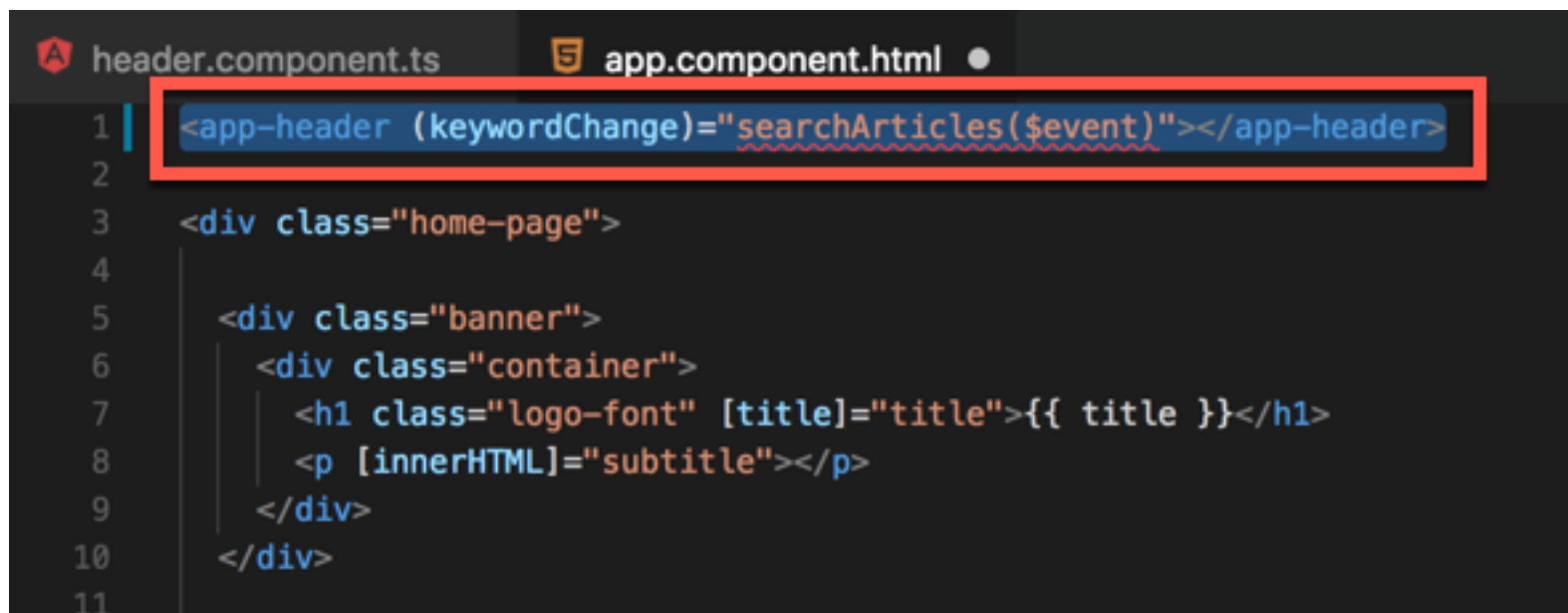
任務10：使用@Output()接受元件輸出的資料

- 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 x
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.html 中使用事件綁定取得 keywordChange 的事件資料
 - `<app-header (keywordChange)="searchArticles($event)"></app-header>`



```
header.component.ts  app.component.html
1 | <app-header (keywordChange)="searchArticles($event)"></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
```

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;  
    }  
  }  
}
```



Introduction

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

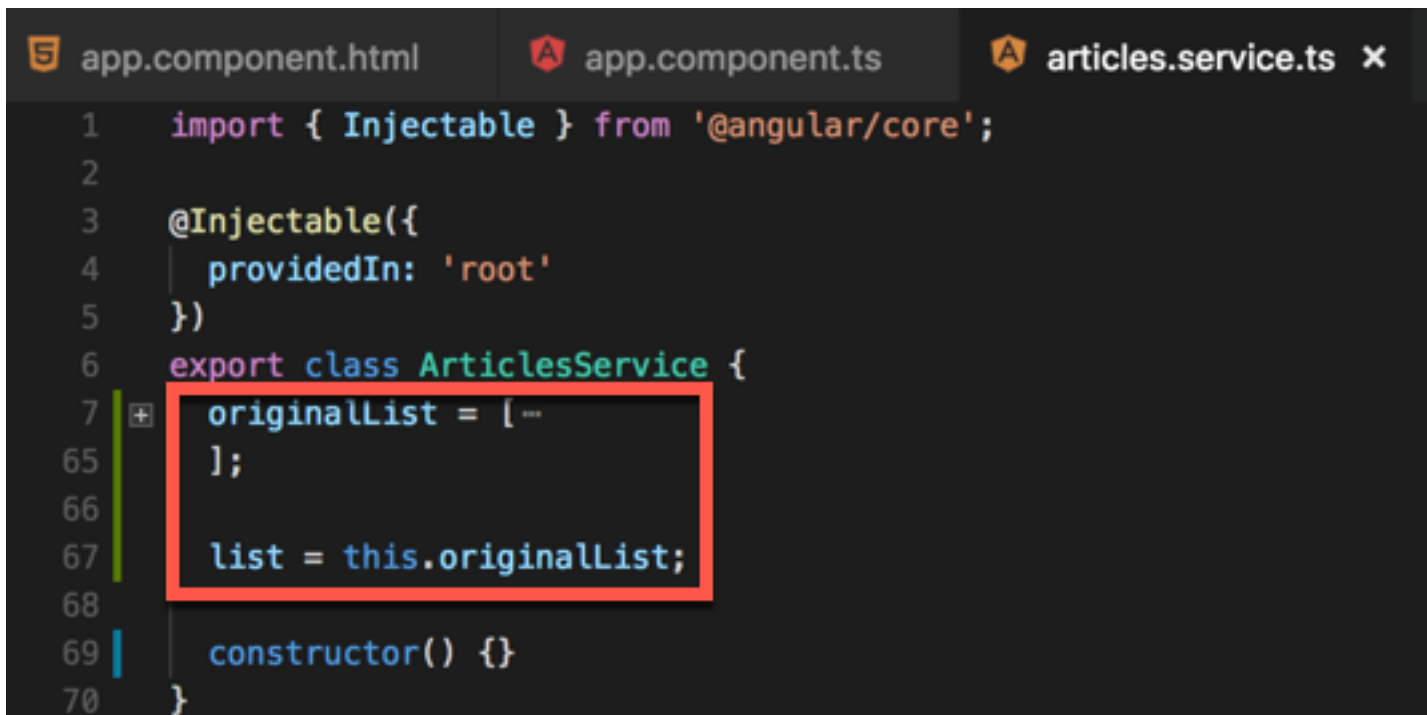
任務11：自訂服務元件

- 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

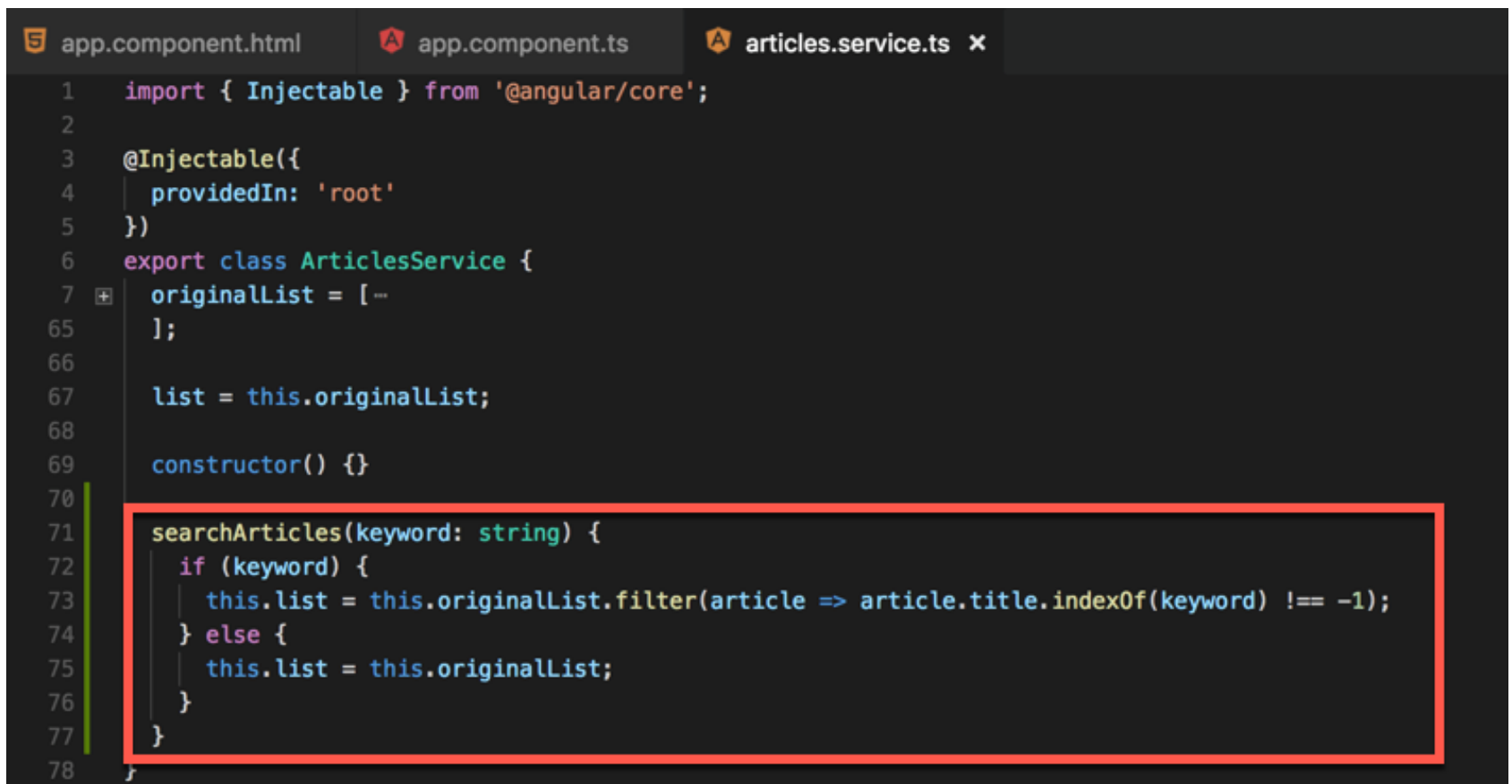
- 將文章相關的資訊移動到 ArticlesService 中



```
app.component.html  app.component.ts  articles.service.ts x
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 x
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
71    searchArticles(keyword: string) {
72      if (keyword) {
73        this.list = this.originalList.filter(article => article.title.indexOf(keyword) !== -1);
74      } else {
75        this.list = this.originalList;
76      }
77    }
78  }
```

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



The screenshot shows the code for header.component.ts. The top part shows imports and the @Component decorator. The bottom part shows the constructor and the search method. Two red boxes highlight the constructor parameter and the method call, with red arrows pointing to them and Chinese text annotations.

```
header.component.ts x
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
12   constructor(private articlesService: ArticlesService) {}
13
14   ngOnInit() {}
15
16   search($event) {
17     this.articlesService.searchArticles(this.keyword);
18   }
19 }
```

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 }
19
```

```
articles.service.ts
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class ArticlesService {
7   originalList = [
8     ...
9   ];
10
11   list = this.originalList;
12
13   constructor() {}
14
15   searchArticles(keyword: string) {
16     if (keyword) {
17       this.list = this.originalList.filter(article => ar
18     } else {
19       this.list = this.originalList;
20     }
21   }
22 }
```

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？

```
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 }
19
```

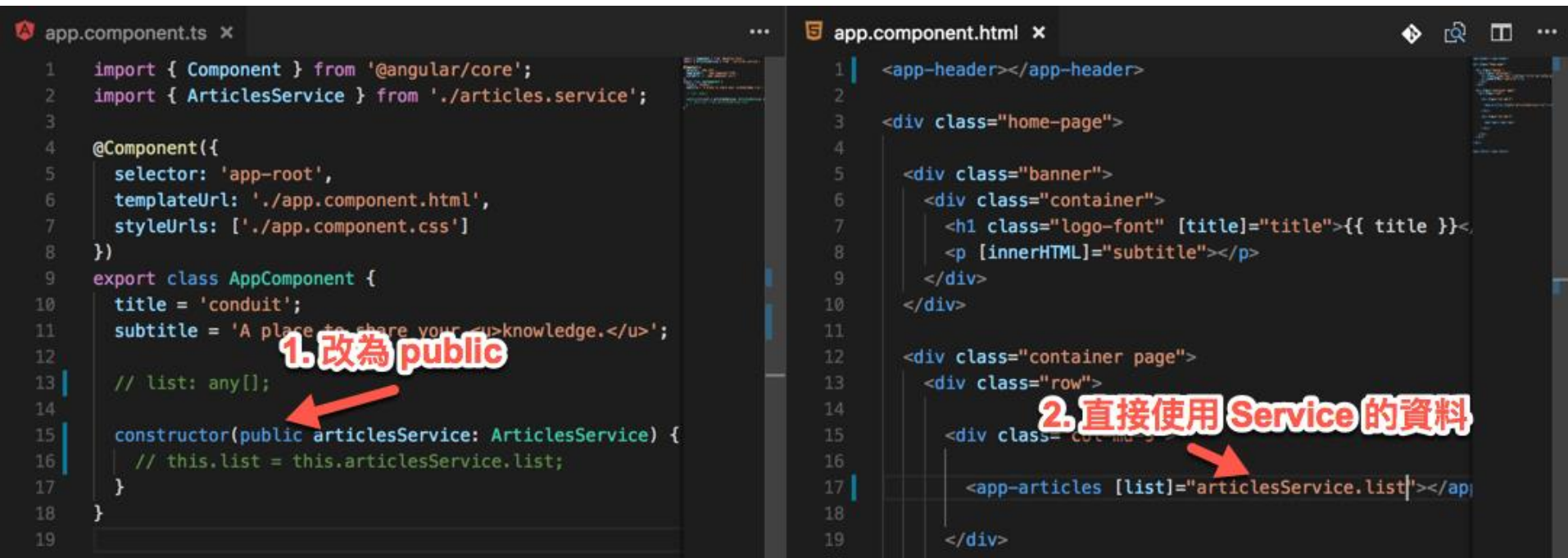
但是AppComponent的list資料沒有跟上

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

ArticlesService的list資料更改了

任務11：自訂服務元件

- 修正方法1：讓 ArticlesService 變成 public



The image shows two side-by-side code editors in VS Code. The left editor, titled 'app.component.ts', shows the TypeScript code for the AppComponent. The right editor, titled 'app.component.html', shows the HTML template for the component. Red annotations with arrows highlight the changes made to make the service public and use its data directly.

```
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(public articlesService: ArticlesService) {
16     // this.list = this.articlesService.list;
17   }
18 }
19
```

1. 改為 public

```
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" [title]="title">{{ title }}<
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-8">
16
17         <app-articles [list]="articlesService.list"></app-articles>
18
19       </div>
20     </div>
21   </div>
22
```

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  
  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

```
app.module.ts x
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],
14    providers: [],
15    bootstrap: [AppComponent]
16  })
17  export class AppModule {}
18
```

任務12：使用 HttpClient 服務

- 在 ArticlesService 注入 HttpClient 並呼叫 API

```
@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: 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](#)



Introduction

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`



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">
      <i class="ion-heart"></i> {{ article.likesCount }}
    </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 (points to `date`)

Pipe參數 (points to `'yyyy-MM-dd'`)

補充：好用的 AsyncPipe

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

```
export class AppComponent implements OnInit {  
  title = 'conduit';  
  subtitle = 'A web application for managing articles';  
  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

- 在 app.component.html 中，使用 AsyncPipe，幫助我們處理非同步資料

```
<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 x ★ 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

```
第一個參數代表傳入的資料  
})  
第三個代表Pipe的參數  
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) => 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



Introduction

補充内容

TypeScript

- 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

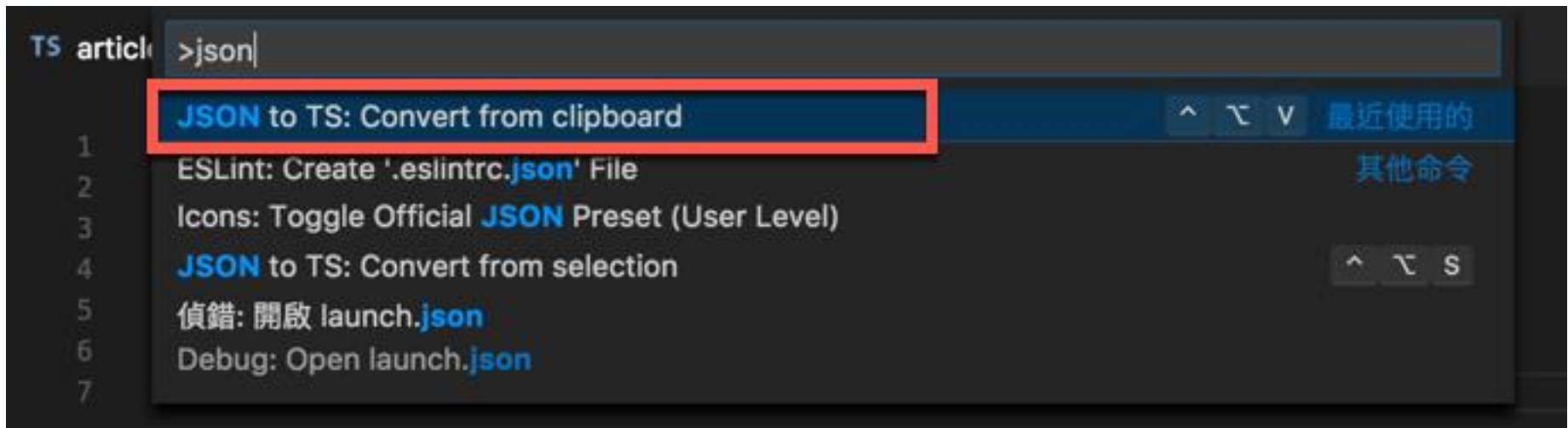
- 使用 Angular CLI 產生 interface
 - `ng g i article`
 - 打開 `article.ts`
 - 定義屬性類型

A screenshot of a code editor window titled "TS article.ts" with a close button. The editor shows a TypeScript interface definition for "Article". The code is as follows:

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

TypeScript

- 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

- 在VS Code 宣告型別後，可以體驗到強型別的好處

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

  @Input() list: Article[];

  constructor() { }

  ngOnInit() {
    this.list[0].
  }
}
```

因為宣告型別的關係，
相關屬性可以 auto complete

- description
- favoritesCount
- title (property) Article.title: string

RxJS

- 在 Angular 中，幾乎所有的非同步行為都是透過 RxJS
- 概念類似 Promise
- 加入資料流的概念
- 使用 **Subscribe()** 來執行內容並取得結果
 - 類似 Promise 的 **then()**

RxJS

- 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



Introduction

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](#)