多奇·教育訓練

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

主軸1：Angular 專案基礎架構

說明

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`

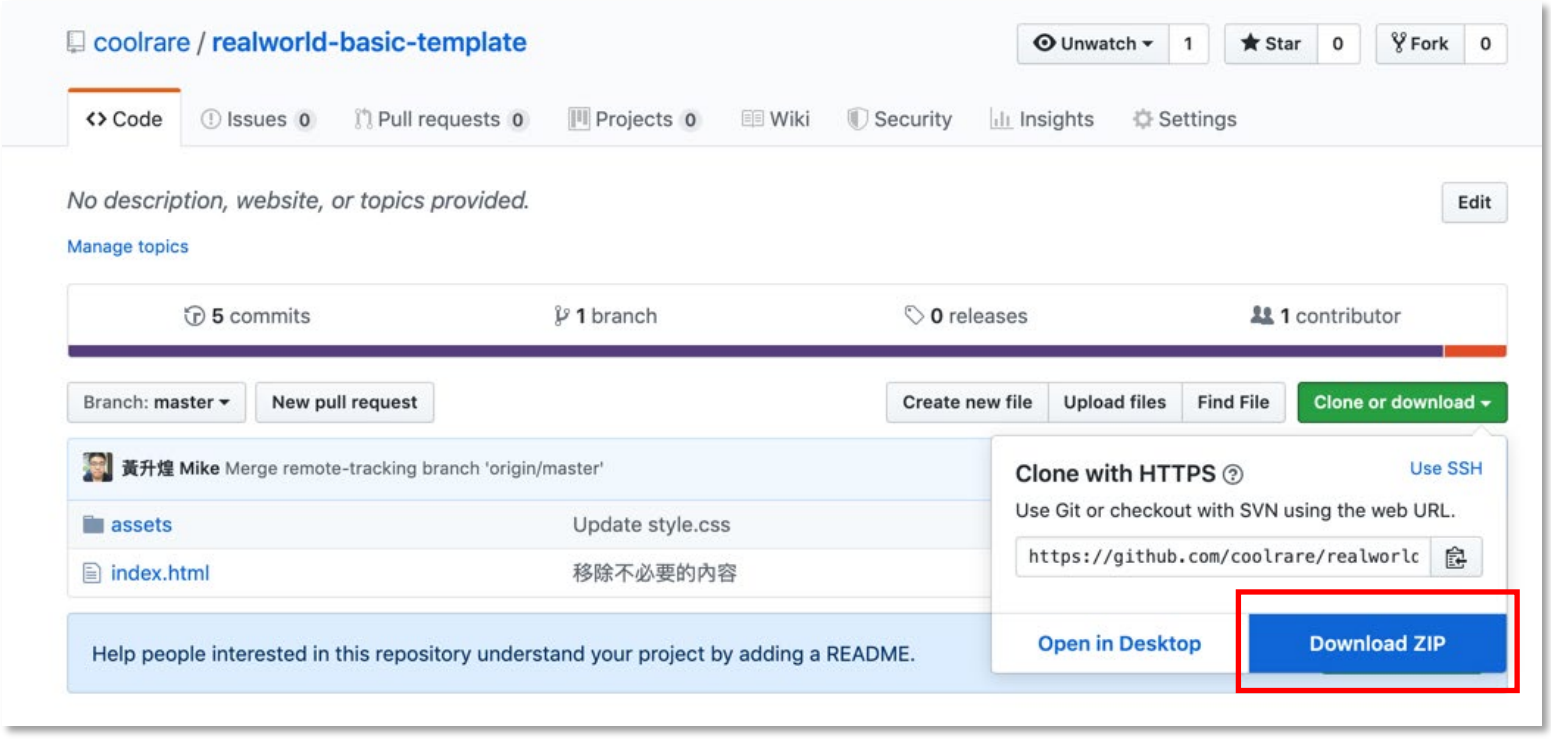
補充：使用 yarn 作為套件管理器

- `ng set --global packageManager=yarn`

說明

任務02：現有模板切版，及元件規劃

- 下載模板



說明

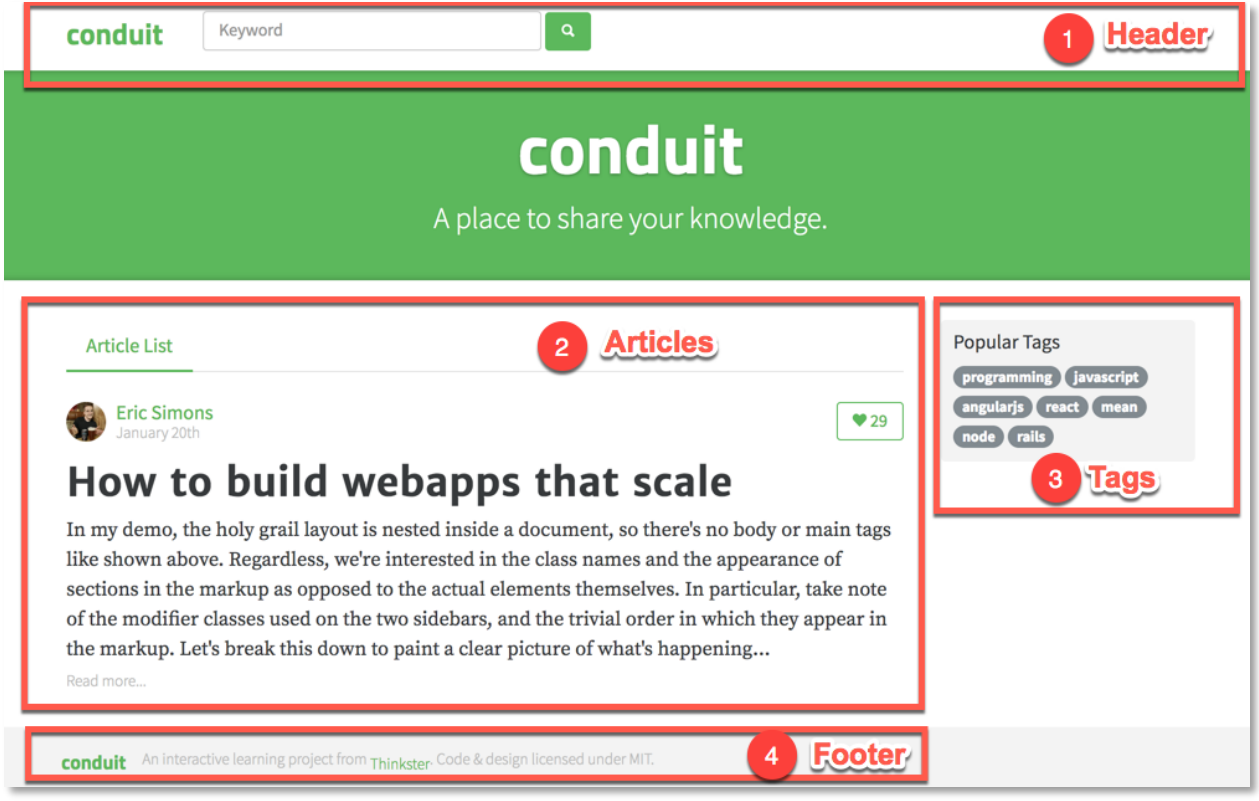
任務02：現有模板切版，及元件規劃

- 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>

說明

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



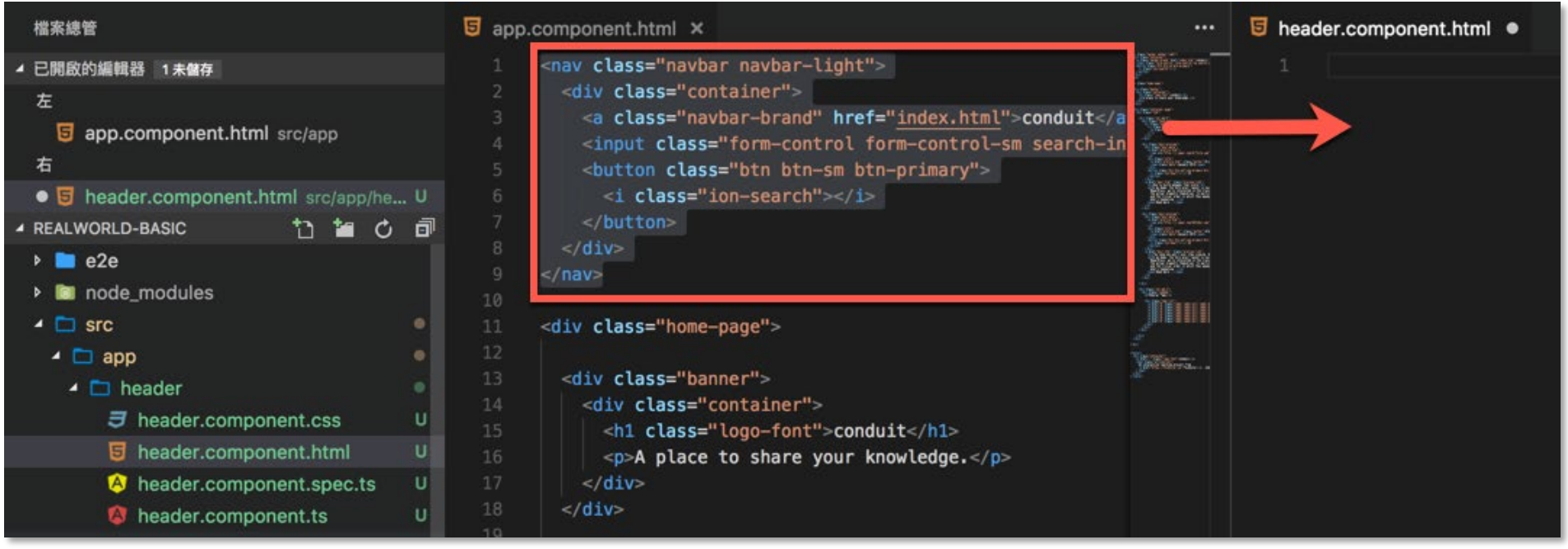
任務02：現有模板切版，及元件規劃

- 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)
```

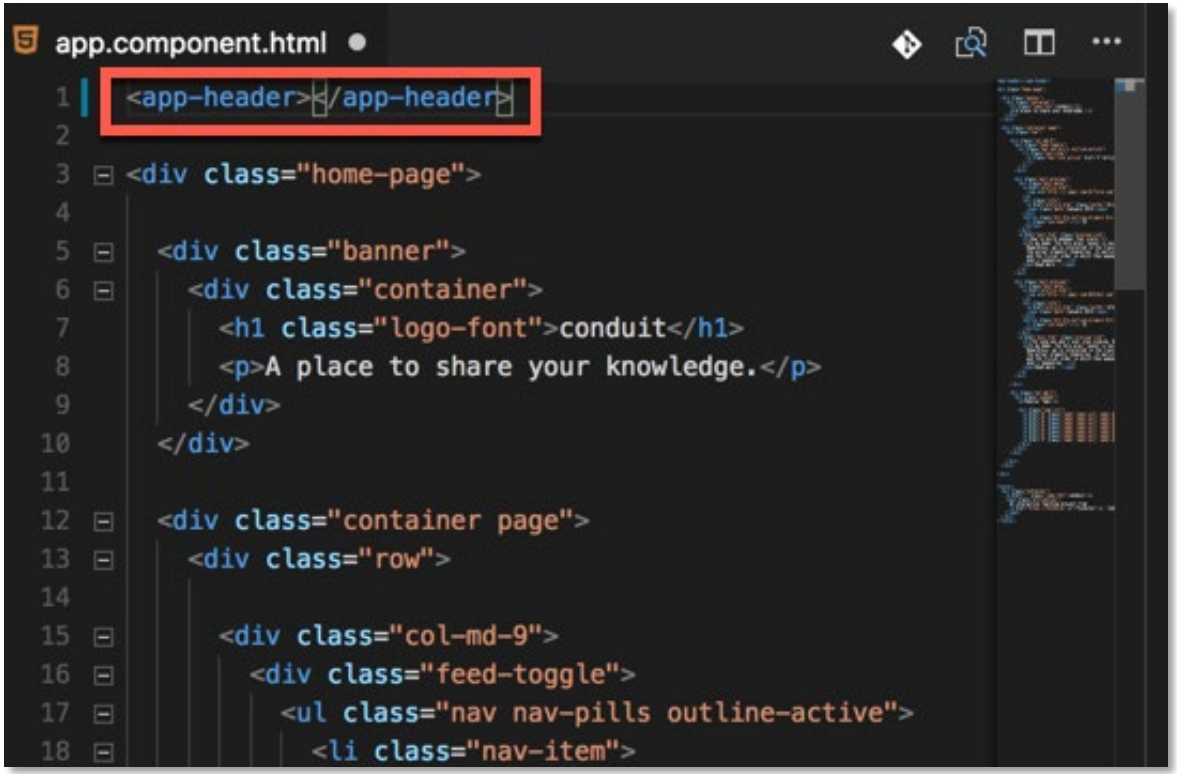
任務02：現有模板切版，及元件規劃

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



補充：如果專案不需要測試檔(*.spec.ts)

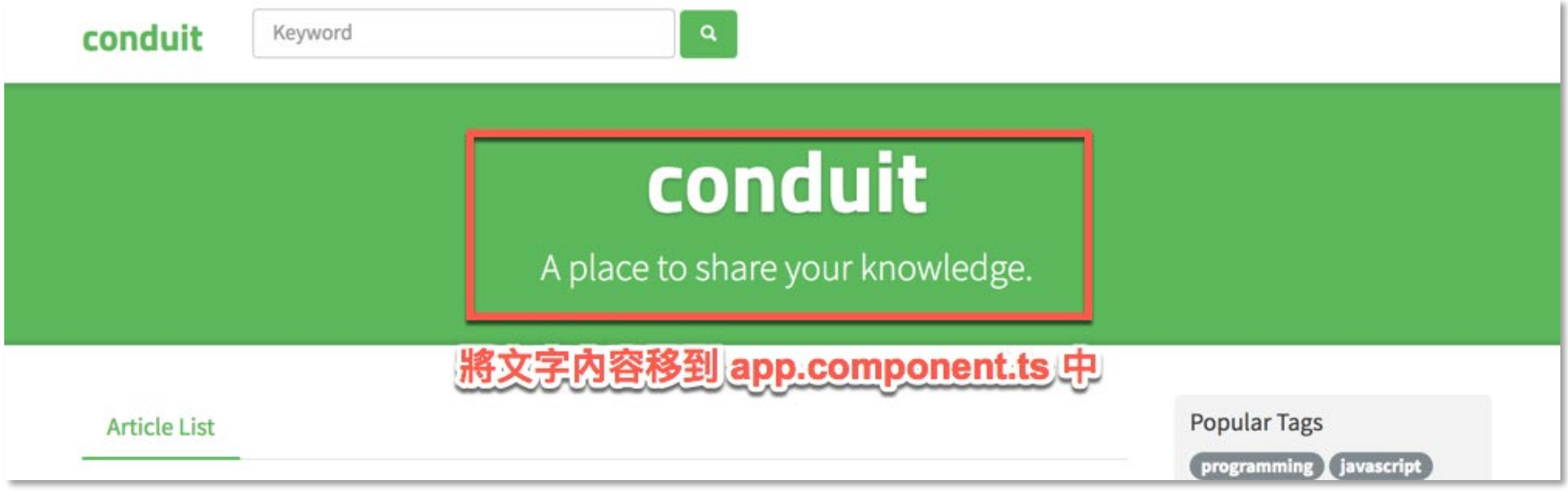
- 方法1：
  - `ng new [project name] --skip-tests`
- 方法2：
  - `ng g c [component name] --spec=false`
- 方法3：在 angular.json 中進行統一設定
  - `projects.[project name].schematics`
  - 參考程式碼

主軸2： Angular 畫面呈現與應用

說明

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



任務03：使用內嵌繫結顯示資料

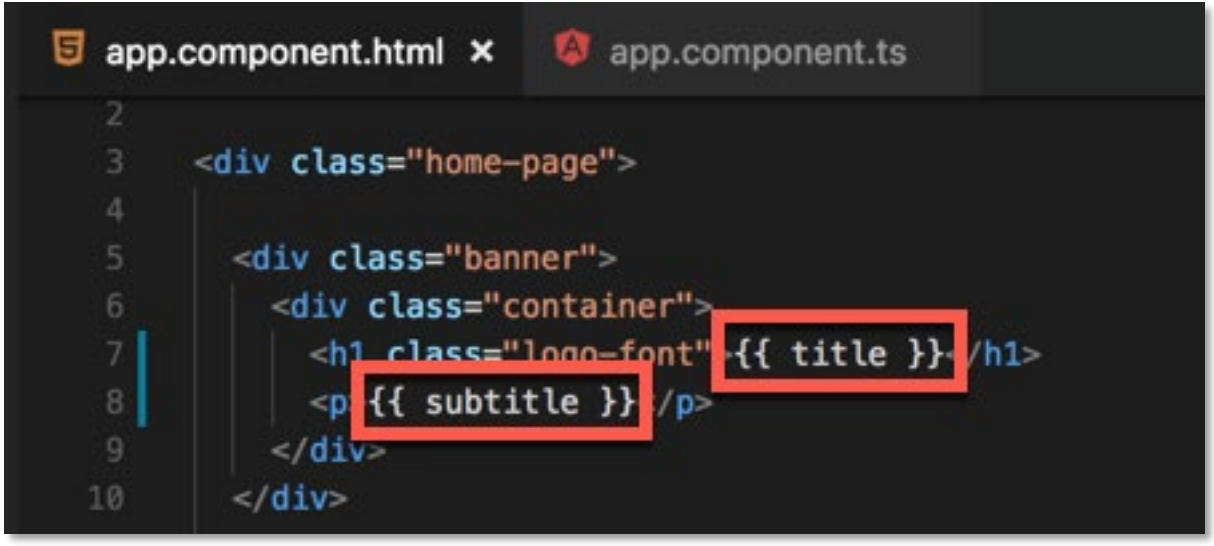
- 在 app.component.ts 中加入 title 與 subtitle 屬性

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

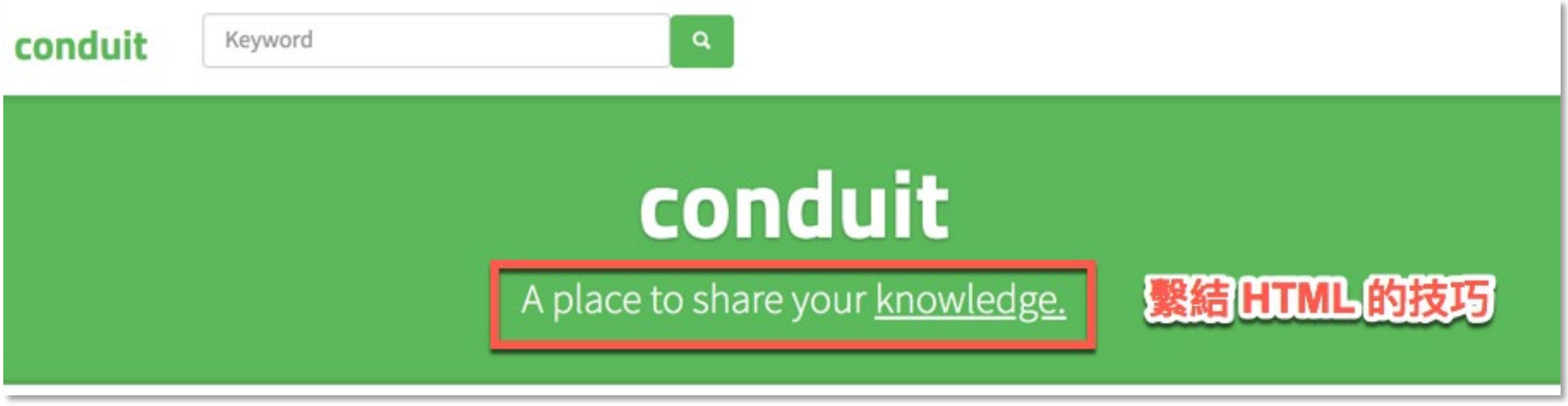
- 在 app.component.html 中繫結屬性



說明

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

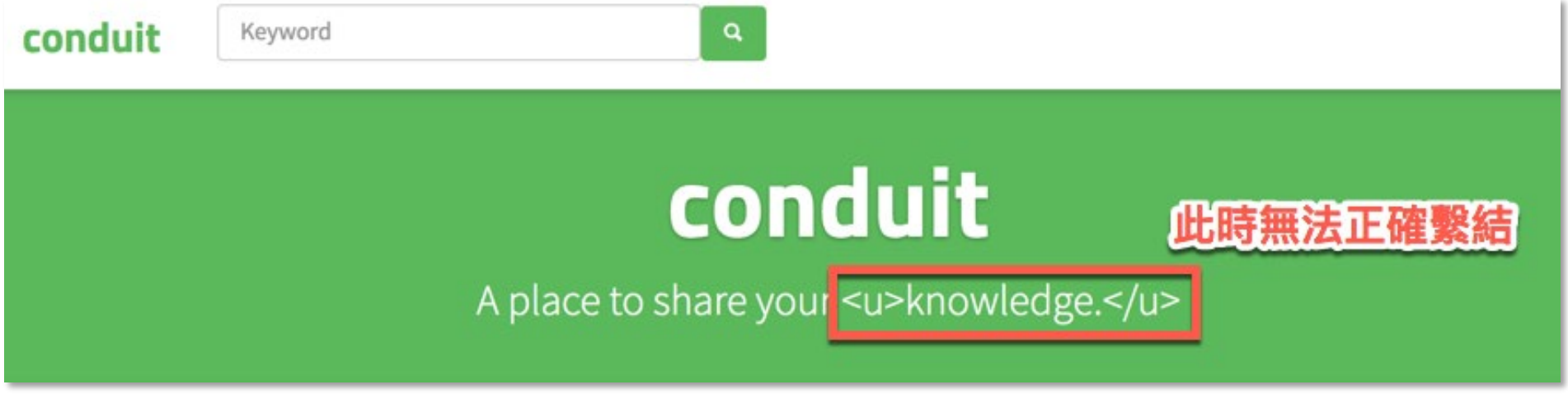


任務04：使用屬性繫結顯示資料

- 使用 `[title]="title"` 繫結資料

任務04：使用屬性繫結顯示資料

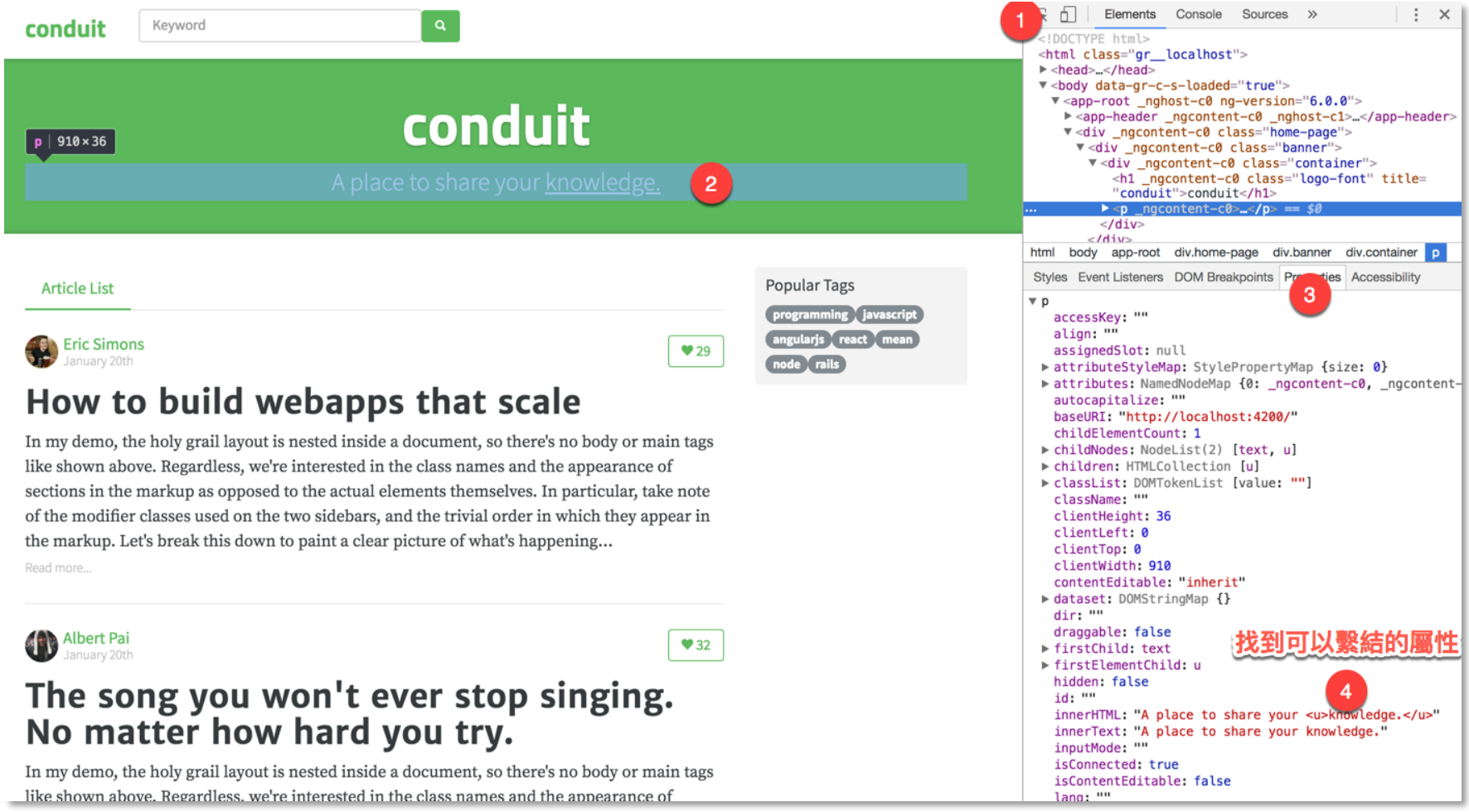
- 將 `subtitle` 改為包含 HTML 的資料



任務04：使用屬性繫結顯示資料

- 使用屬性繫結：[innerHTML]="subtitle"

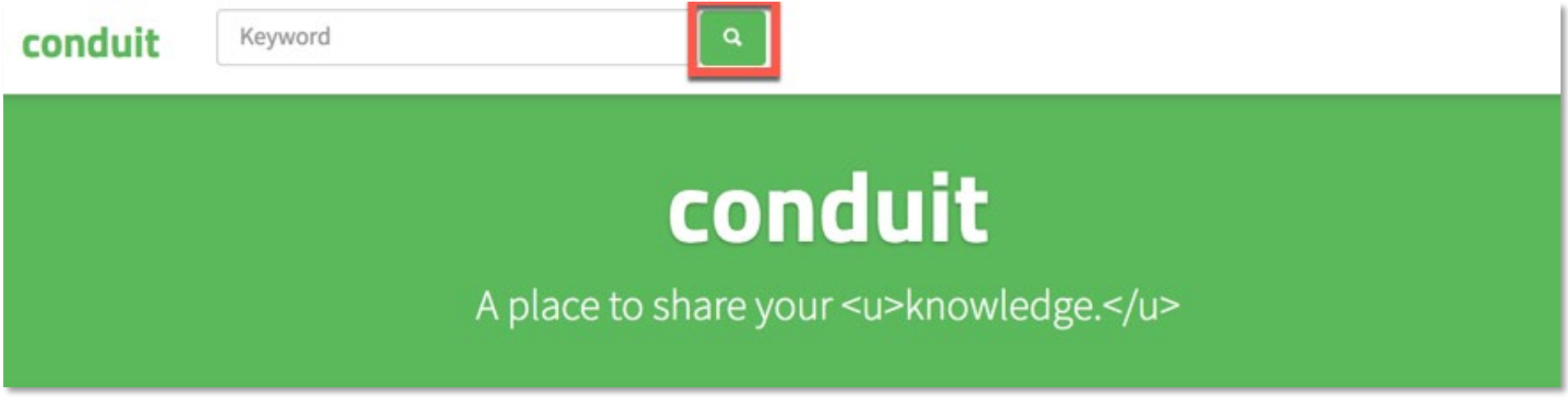
補充：如何知道哪些屬性可以繫結



說明

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

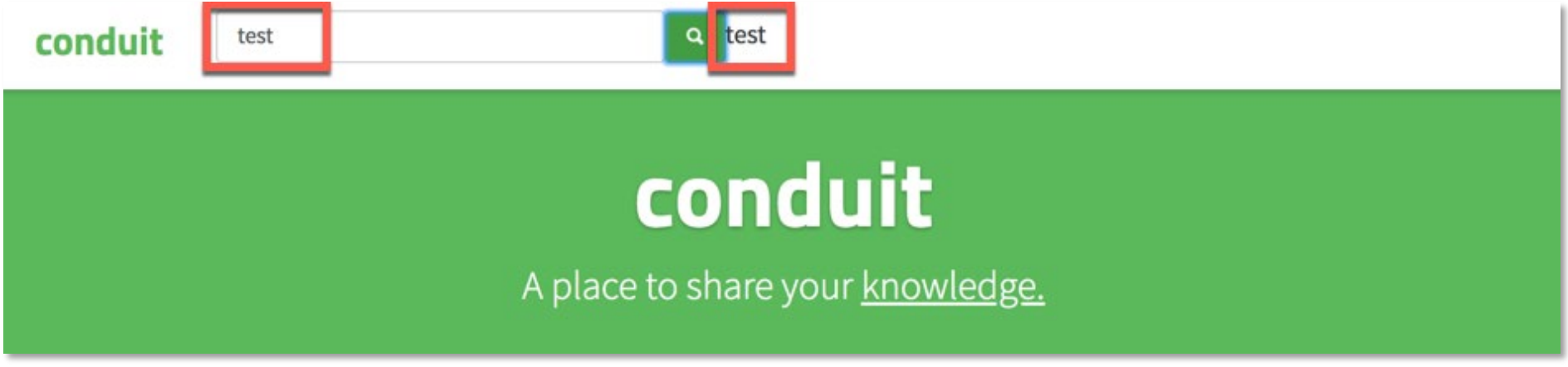
header.component.html　header.component.ts ×

```
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 }
```

說明

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>
```

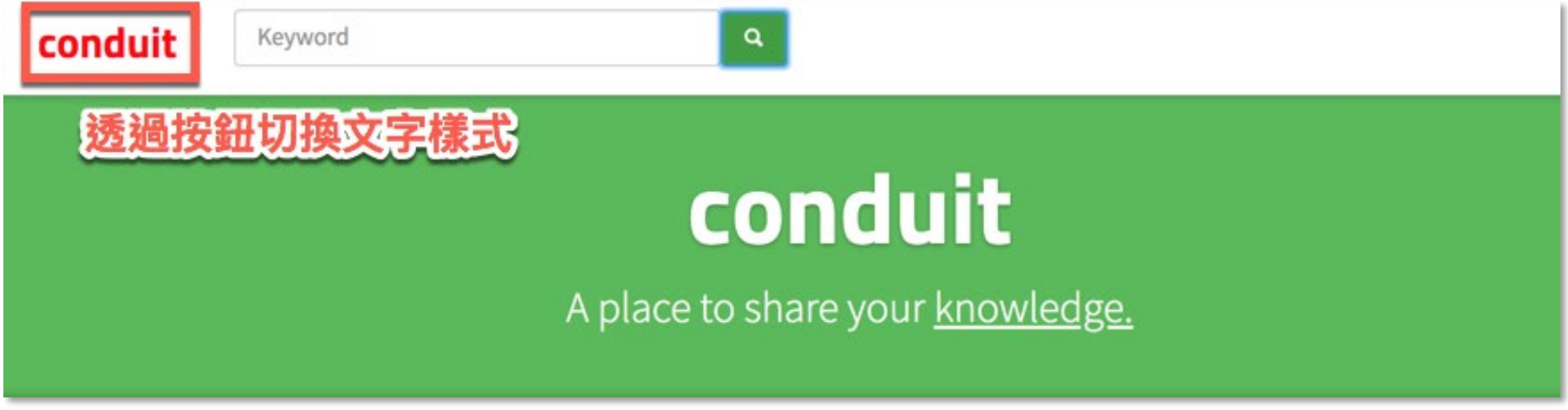
補充：雙向繫結的原理

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

說明

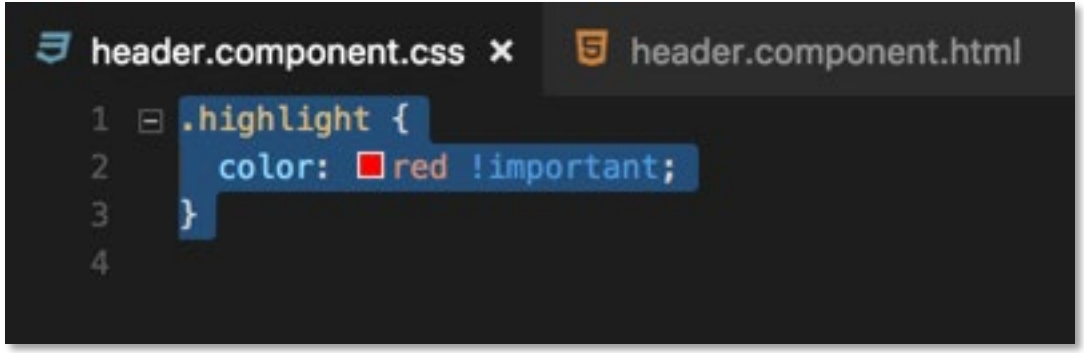
任務07：使用屬性指令(Attribute Directive)

- 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



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.highliht]="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.font-size.px]="fontSize"

說明

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)

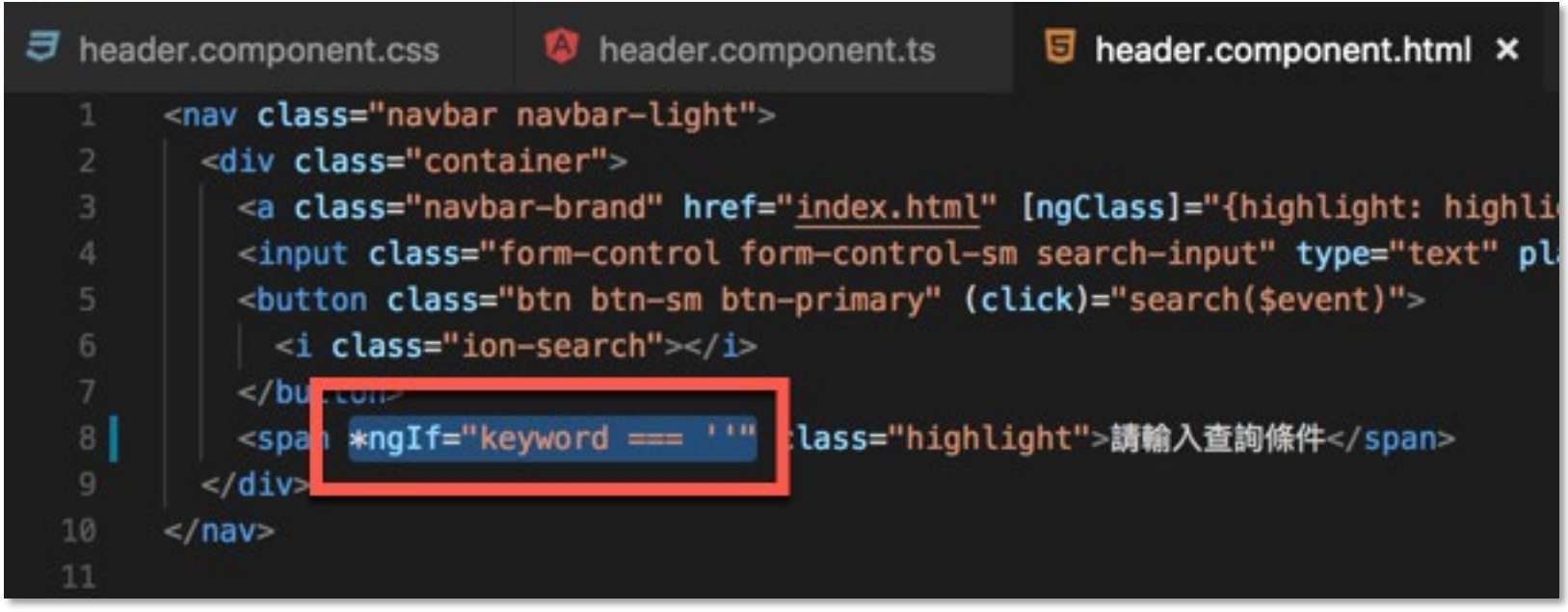
- 在 header.component.html 中加入提示文字

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: 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
```

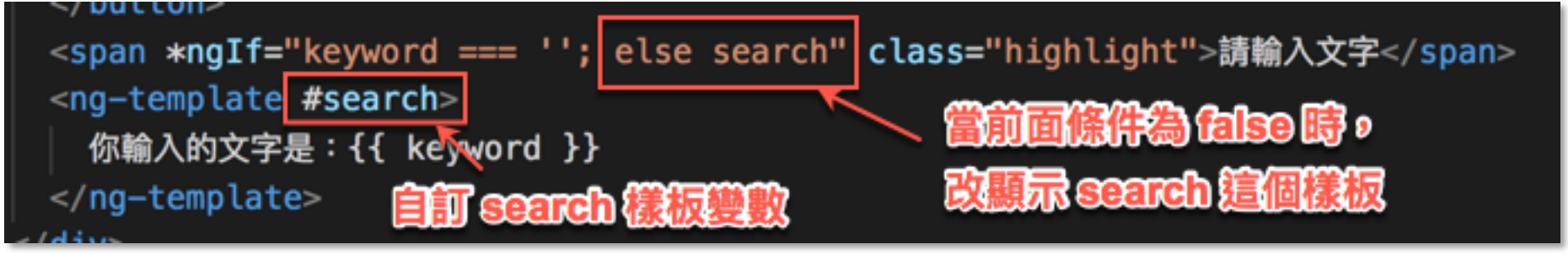
任務08：使用結構指令(Structural Directive)

- 套用 *ngIf 判斷是否有輸入查詢條件



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**



說明

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

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 necessita ibus\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 ne essitatibus\ndolor qua
      "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

說明

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 reprehen
      "slug": "zp7yqc",
      "body": "laudantium enim quasi est quidem magnam voluptate ipsam eos\ntempora q
      "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\nte
      "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變變數改成
  - `@Input() list: any[];`

articles.component.ts × | 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() { }
```

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

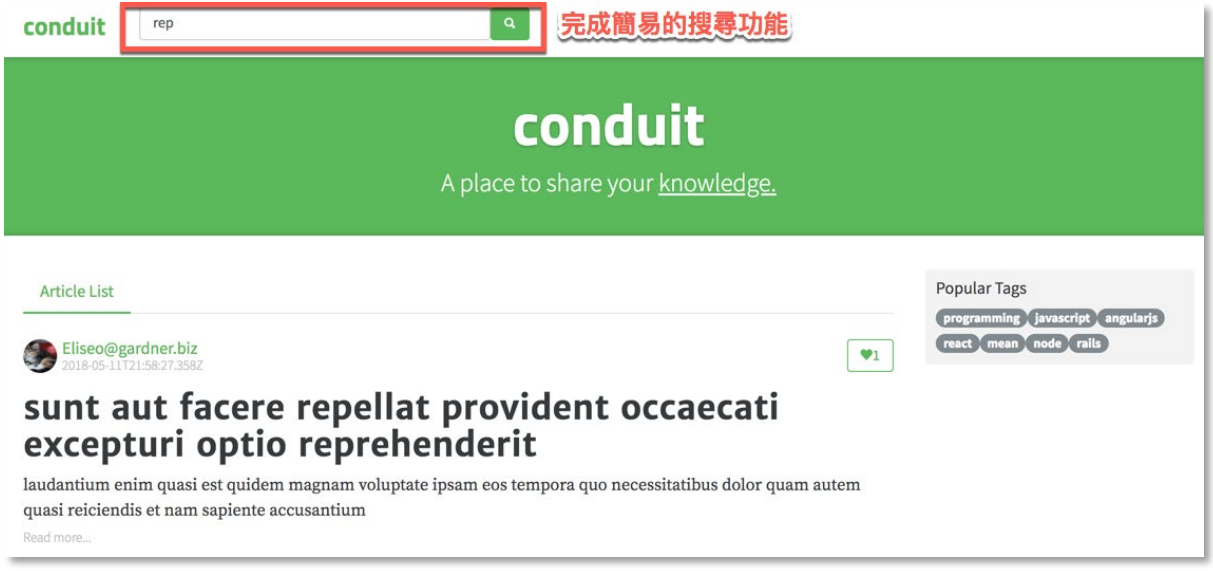
@Input() 變成元件的屬性

app.component.ts 中的變數名稱

說明

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

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

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

  @Output() keywordChange = new EventEmitter<string>();

```

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

```ts
import { Component, OnInit, EventEmitter, Output } from '@angular/core';

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

  @Output() keywordChange = new EventEmitter<string>();

  keyword = '';
```

```ts
  search($event) {
    this.keywordChange.emit(this.keyword);
  }
}
```

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

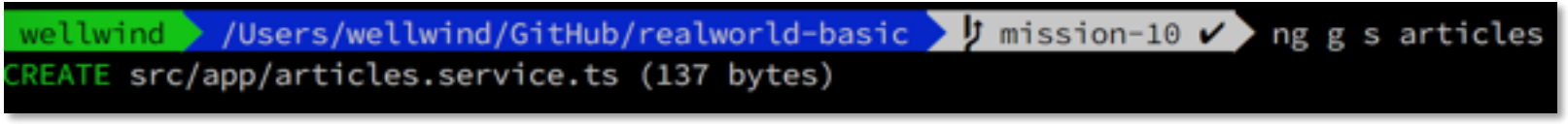
說明

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



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 = […
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

任務11：自訂服務元件

- 在 app.component.ts 中，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

```ts
import { Component, EventEmitter, OnInit, Output } from '@angular/core';
import { ArticlesService } from '../articles.service';

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

```ts
  constructor(private articlesService: ArticlesService) {}

  ngOnInit() {}

  search($event) {
    this.articlesService.searchArticles(this.keyword);
  }
}
```

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

```ts
import { Component } from '@angular/core';
import { ArticlesService } from './articles.service';

@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: any[];

  constructor(private articlesService: ArticlesService) {
    this.list = this.articlesService.list;
  }
}
```

ArticlesService 的 list 資料更改了

articles.service.ts

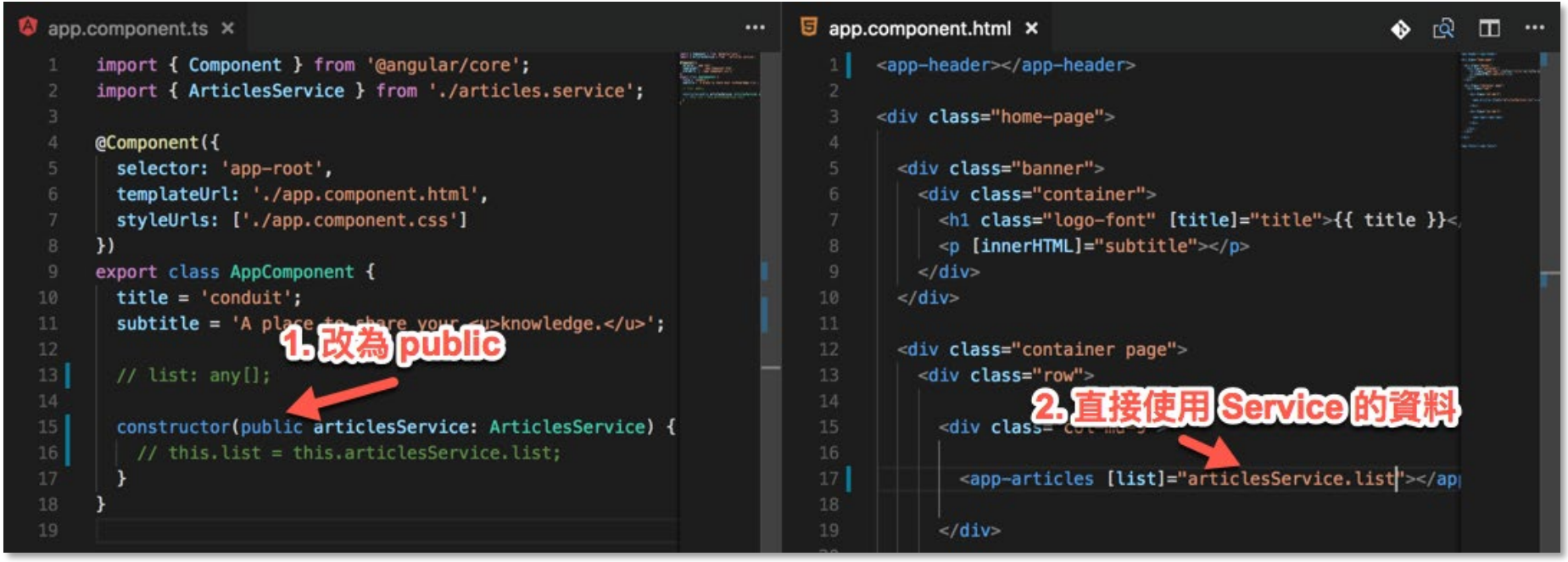
```ts
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class ArticlesService {
  originalList = [...
  ];

  list = this.originalList;

  constructor() {}

  searchArticles(keyword: string) {
    if (keyword) {
      this.list = this.originalList.filter(article => ar
    } else {
      this.list = this.originalList;
    }
  }
}
```

任務11：自訂服務元件

- 實際測試時，沒有效果，為什麼？

任務11：自訂服務元件

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



任務11：自訂服務元件

- 修正方法2：改用 getter

說明

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

```
import { HttpClientModule } from '@angular/common/http';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { ArticlesComponent } from './articles/articles.component';
import { FooterComponent } from './footer/footer.component';
import { HeaderComponent } from './header/header.component';
import { TagsComponent } from './tags/tags.component';

@NgModule({
  declarations: [AppComponent, HeaderComponent, FooterComponent, ArticlesComponent, TagsComponent],
  imports: [BrowserModule, FormsModule, HttpClientModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

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

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

說明

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`

任務13：使用Angular內建的Pipe

- 在 ArticlesCompnent 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.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 Obserable 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

說明

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

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((respons
      this.list = response.articles;
    });
  }

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

  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)

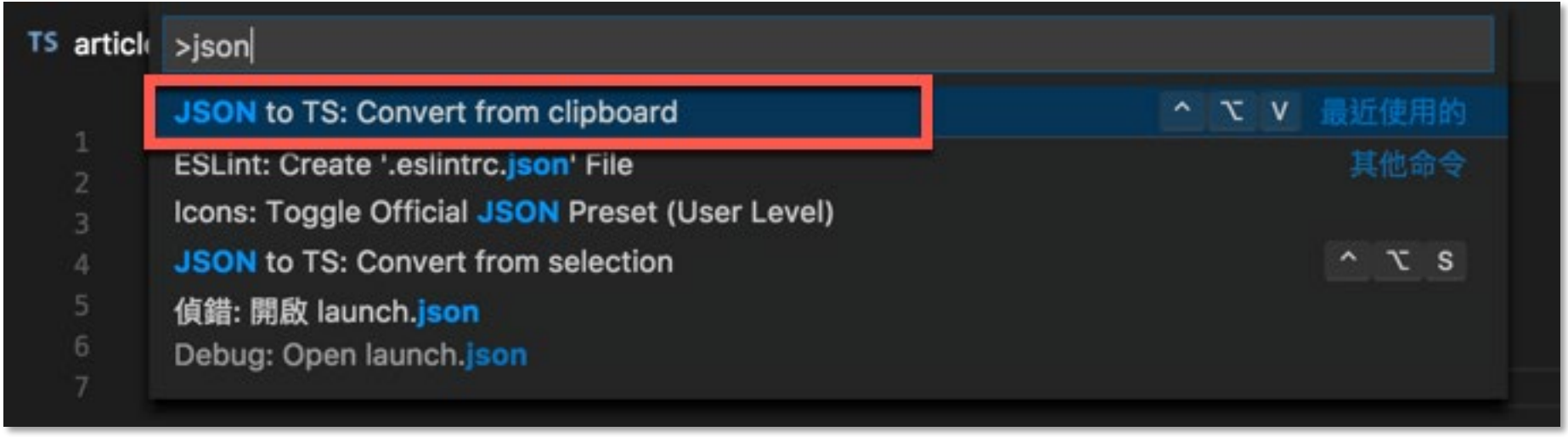
- 使用 Angular CLI 產生 interface
  - `ng g i article`
  - 打開 `article.ts`
  - 定義屬性類型

TS article.ts ×

```ts
export interface Article {
  title: string;
  description: string;
  favoritesCount: number;
}
```

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

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

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