« Angular 2 開發實戰 »

新手入門篇

(適用 Angular 2.0.0-final 版本)

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/

Angular 2: Introduction

ANGULAR 2 簡介

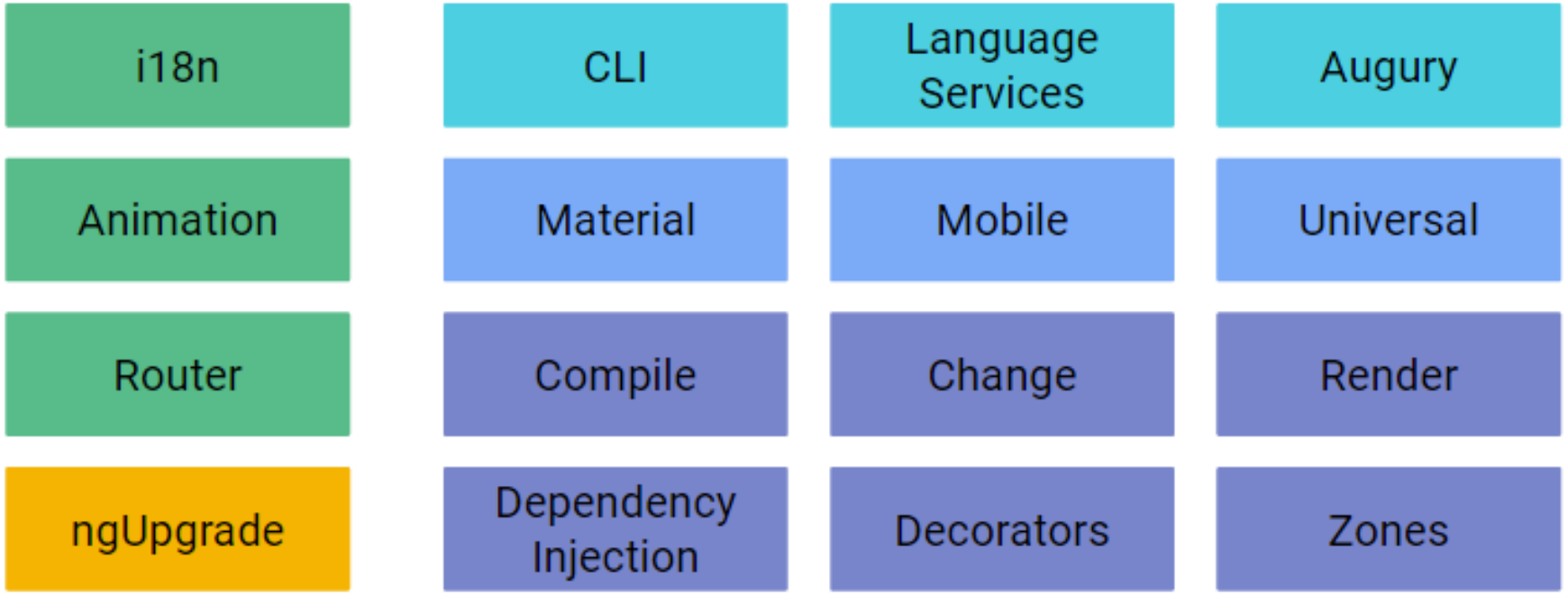
何謂 AngularJS 框架

- 全球領先的開源 JavaScript 應用程式框架
  - 由 Google 主導並擁有廣大社群共同參與框架發展

- Angular 1.x
  - 擁有廣大開發社群 (最大的)
  - 透過嶄新的抽象化架構大幅簡化應用程式開發

- Angular 2.x
  - 重新打造的下一代 AngularJS 開發框架
  - 擁有更高的執行效率、更好的延展性架構
  - 透過全新的元件化技術建構現代化的開發框架

從框架轉向平台



[前端工程的夢幻逸品：Angular 2 開發框架介紹]

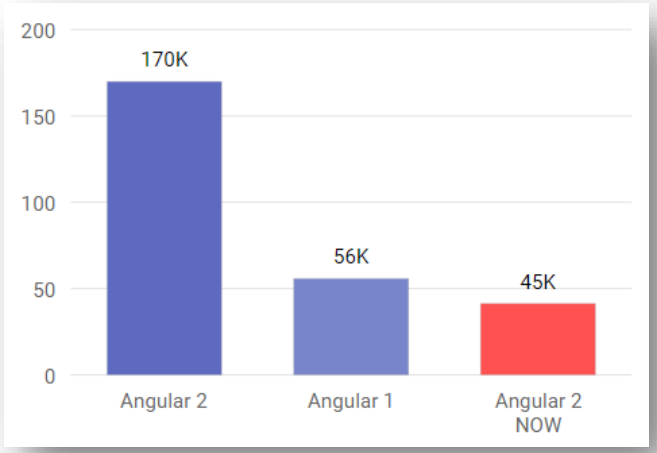
Angular 2 主要特色 (1)

- 跨平台
  - Progressive Web Apps（Angular Mobile Toolkit）
    - 結合**網頁**和**應用程式**優點於一身的絕佳體驗
  - Desktop Apps
    - 可搭配 Electron 框架開發出跨越 Windows, Mac, Linux 的桌面應用程式
  - Native Apps
    - 可搭配 Ionic Native, NativeScript, React Native 開發跨行動平台的原生應用程式
- 速度與效能
  - Code generation (AOT)
    - 將元件範本預先編譯成 JS 程式碼
  - Universal
    - 將開啟頁面的首頁預先產生完整 HTML 與 CSS 原始碼，加快首頁載入速度
    - 可支援 Node.js, .NET, PHP 或任何其他伺服器端網頁架構
  - Code Splitting
    - 透過全新的元件路由機制，讓使用者只須載入需要的原始碼

Angular 2 主要特色 (2)

- 生產力提升
  - Templates
    - 使用簡易有強大的範本語法提高開發效率
  - Angular CLI
    - 透過命令列工具快速建模、新增元件、執行測試與發行部署
  - IDE
    - 在現有編輯器或開發工具中使用程式碼自動完成、即時錯誤提示與程式碼建議
- 完整的開發體驗
  - Testing
    - 結合 Karma 執行單元測試，結合 Protractor 執行各種 E2E 測試情境
  - Animation
    - 透過 Angular 直觀的 API 完成複雜的頁面動畫處理
  - Accessibility
    - 透過 ARIA-enabled 元件、開發者指引與內建的 a11y 測試基礎架構，建構具有可及性的應用程式

Angular 2 主要亮點

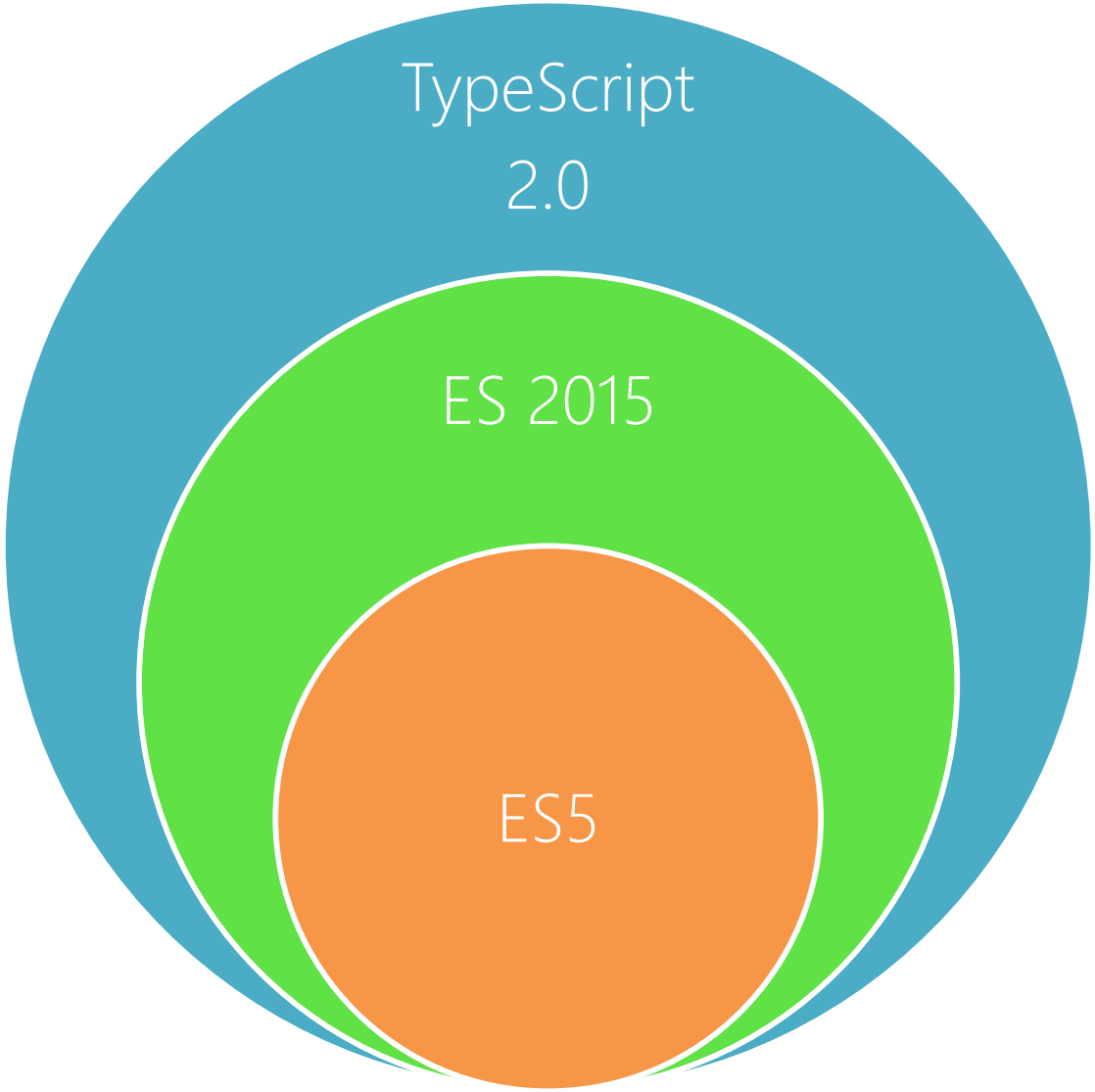


- 效能改進 (Performance)
  - 偵測變更：比 ng1 快 10 倍
  - 更小的 Library Size 與延遲載入機制
  - 範本編譯：支援 Template 預先編譯機制
  - 渲染速度：比 ng1 快 5 倍 (Render & Re-render)
  - 支援伺服器渲染機制 (Node.js & ASP.NET)
- 高生產力 (Productivity)
  - 開發應用程式能夠用更簡潔的語法讓團隊更加容易上手跟維護
  - 更強大的開發工具 Augury
  - 移除超過 40+ 個 directives
- 多樣平台 (Versatility)
  - 支援 Browser, Node.js, NativeScript, and more …

Angular 2 真正優勢

- 更熟悉的開發架構
  - 採用 TypeScript 開發語言，使用以類別為基礎的物件導向架構開發 Web 應用程式，幫助 C#, Java, PHP, …等開發人員快速上手全新架構。
  - 透過開發人員手邊現有的開發工具/編輯器，就可以開發 Angular 2 應用程式，並同時享有 IntelliSense、程式碼重構等工具支援。
- 更低的學習門檻
  - 相較於 Angular 1 減少了許多抽象的架構與概念，對於剛入門的 Angular 開發者將更加容易上手
    - 例如 Angular 1 的 directives 就有非常多抽象概念
- 更好的執行效率與行動化體驗
  - 不同行動裝置之間的各種特性皆考量在內，例如觸控、螢幕大小、硬體限制、…
  - 內建伺服器渲染技術 (server rendering) 與 Web Worker 技術改善頁面載入效率
  - 不僅僅做到預先產生 HTML 頁面，更能透過 NativeScript 或 Ionic 建立起網站框架與 Mobile App 之間的橋樑，開發速效率更好的行動瀏覽體驗。
- 更清晰的專案結構與可維護性
  - 使用 ES2015 模組管理機制，搭配 webpack 或 SystemJS 等工具即可立刻上手
  - 全新的元件模組化架構，更能夠幫助大家更快的了解程式碼結構，降低維護成本
  - 好的模組化架構更能降低開發工具的開發難度，也更適合開發大型的網站應用

Angular 2 的開發語言



Dart

Angular 2 的開發語言

- ES5
  - 傳統 JavaScript 程式語言 (IE9+)
- ES 2015 (ES6)
  - 此版本為 ES5 的「超集合」
  - 具有新穎的 JavaScript 語言特性 (let, const, for-of, ...)
  - 可透過 Babel 轉譯器將瀏覽器不支援的語法轉為 ES5 版本
- TypeScript
  - 此版本為 ES 2015 的「超集合」
  - 具有強型別特性、內建 ES5 轉譯器 (Transpiler)、更好的工具支援
- Dart
  - 非 JavaScript 家族的程式語言
  - 具有強型別特性

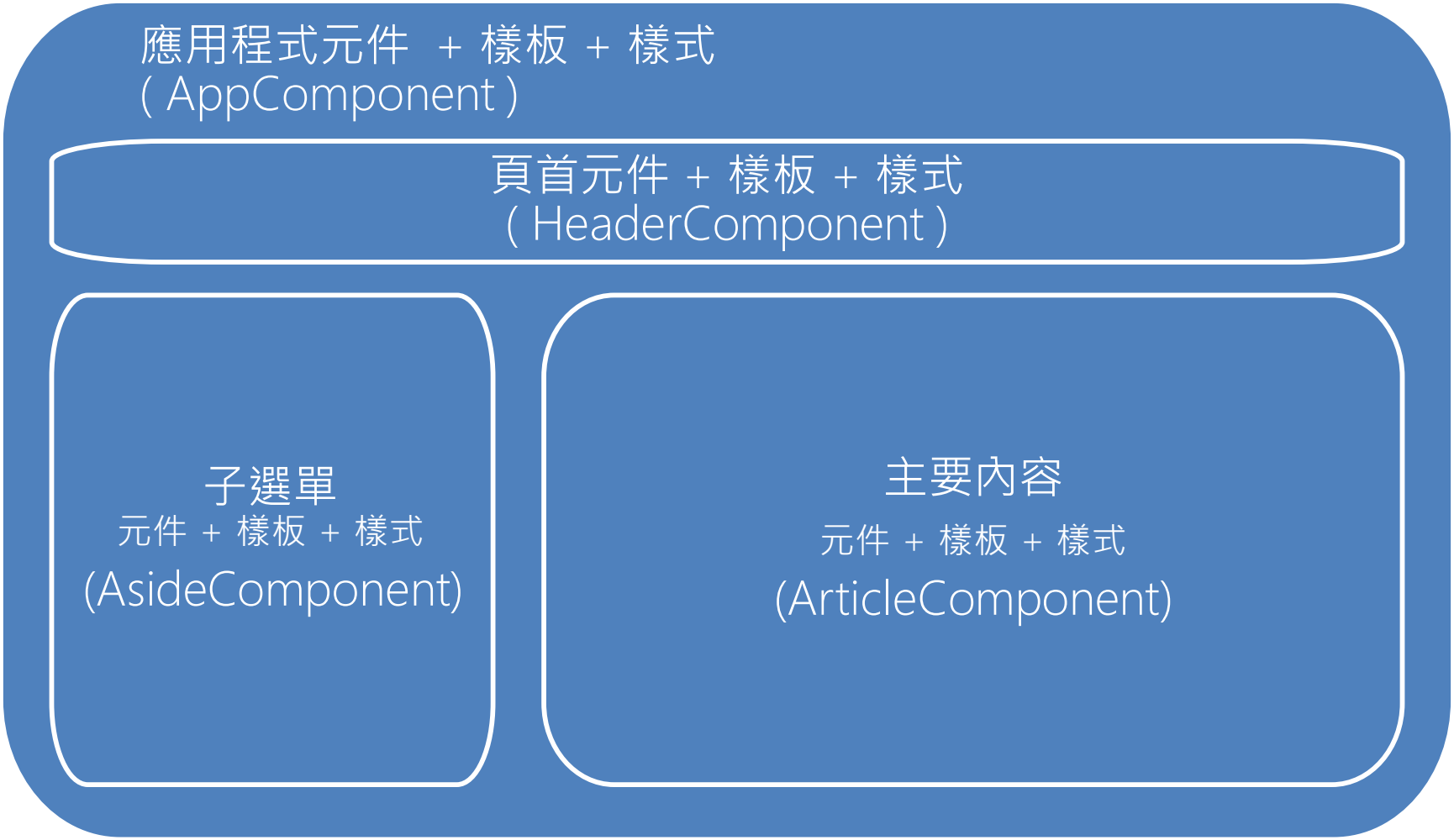
Angular 2 的開發工具

- Visual Studio Code (推薦)
- Visual Studio 2015
- Sublime Text
- WebStorm
- Atom
- Plunker

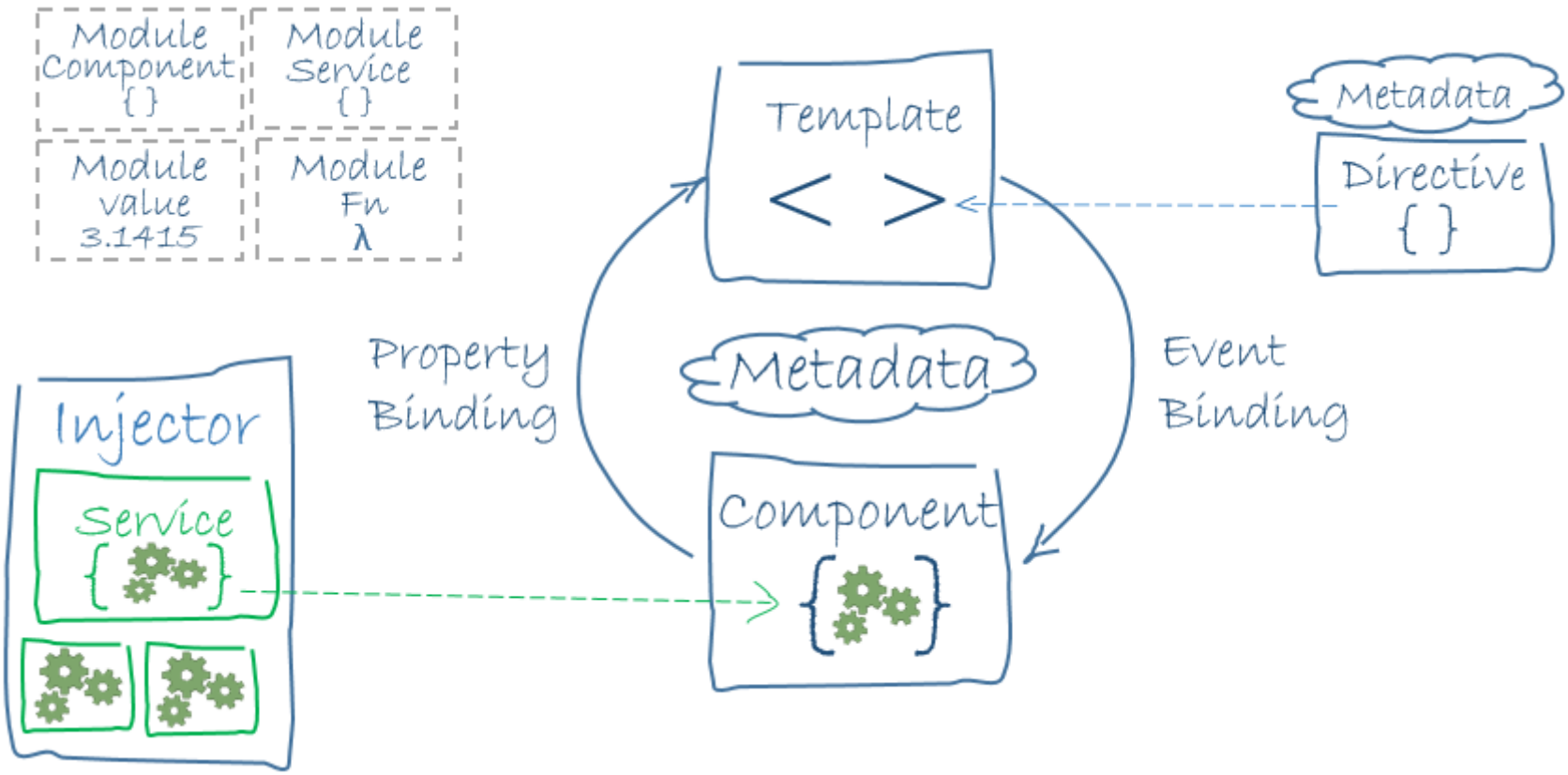
Angular 2 應用程式的組成

- 模組
  - AppModule
    - 元件
      - App Component
    - 元件
      - Child Component
    - 元件
      - Services Component
    - 元件
      - Pipe Component

Angular 2 頁面的組成



Angular 2 結構剖析



Angular 2 結構剖析

- Module　應用程式被切分成許多「模組」
- Component　每個模組下有許多「元件」
- Template　每個元件都可能有自己的「樣板」
- Metadata　每個元件都可以標示「中繼資料」
- Data Binding　樣板與元件屬性、方法可以進行綁定
- Directive　將 DOM 轉換為多功能的「宣告命令」
- Service　由「服務」集中管理資料與運算邏輯
- Dependency Injection　由「相依注入」機制管理物件生命週期

Angular 2 元件的組成

範本 (Template)	類別 (Class)	中繼資料 (Metadata)

• HTML 版面配置
• HTML 部分片段
• 資料繫結 (Bindings)
• 畫面命令 (Directives)
	• 建構式 (Constructor)
	• 屬性 (Properties)
	• 方法 (Methods)
		• 裝飾器 (Decorator)
		  • 針對類別
		  • 針對屬性
		  • 針對方法
		  • 針對參數

認識 Angular 元件的程式碼結構

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

```
@Component({
  selector: 'app-root',
  templateUrl: 'app.component.html',
  styleUrls: ['app.component.css']
})
```

```
export class AppComponent {
  title = 'app works!';
}
```

Setup your development environment

建立 ANGULAR 2 開發環境

準備 Angular 2 開發環境

- 架設 Angular 2 開發環境說明文件
  - Google Chrome、VSCode、Git、Node.js 與 Angular CLI 工具
- 如何修改 Visual Studio Code 內建的 TypeScript 版本
  - Angular 2 預設採用 TypeScript 2.0 為主要開發語言！
  - 但 Visual Studio Code 目前版本還是內建 TypeScript 1.8 版
- 關於 TypeScript 2.0 之後的模組定義檔 (*.d.ts)
  - TypeScrpit 2.0 已改用 npm 來管理模組定義檔
  - 未來不再使用 typings 工具進行模組定義檔管理

安裝 Angular-CLI 建模工具

- **用來快速開發 Angular 2 應用程式的命令提示字元工具**
- 必備條件
  - 安裝 `Node.js 4.x` 以上版本
- 安裝方式
  - `npm install -g angular-cli`
- 升級方法 (Global package)
  - `npm uninstall -g angular-cli`
  - `npm cache clean`
  - `npm install -g angular-cli`
- 升級方法 (local project package)
  - `rm -rf node_modules dist`
  - `npm install --save-dev angular-cli`
  - `ng init` (用來檢查並更新之前透過 `ng new` 自動建立的檔案)
- Angular CLI 版本變更紀錄：CHANGELOG.md

使用 Angular-CLI 建立並執行網站

- 使用說明
  - `ng --help`
- 建立新專案並啟動開發伺服器
  - `ng new PROJECT_NAME`
  - `cd PROJECT_NAME`
  - `ng serve`
  - `http://localhost:4200`
- 指定不同埠號啟動開發伺服器
  - `ng serve --port 4201 --live-reload-port 49153`
- 建立元件 (Components)
  - `ng generate component component1` # 在專案根目錄執行建立元件
  - `ng g component component1` # 可使用 g 簡寫語法 (alias)
  - `ng g c component1` # 可使用 g 簡寫語法 (alias)
  - `ng g c component1` # 在當前目錄下建立元件
  - `ng g c ../component1` # 也可指定相對路徑來建立元件
  - `ng g c component1/subcomp1` # 也可建立特定元件下的子元件

常用 Angular-CLI 命令

Scaffold	使用方式
Component	`ng g component my-new-component`
Directive	`ng g directive my-new-directive`
Pipe	`ng g pipe my-new-pipe`
Service	`ng g service my-new-service`
Class	`ng g class my-new-class`
Interface	`ng g interface my-new-interface`
Enum	`ng g enum my-new-enum`

使用 Angular-CLI 建置與執行

- 建置專案 (預設為 dev 環境)
  - **ng build**
  - 會將現有應用程式建置後輸出到 **dist/** 目錄下
- 建置專案的注意事項
  - 建置專案的過程中，預設為 **dev** 模式 (可切換為 **prod** 模式)
  - 建置專案的過程中，如果是 dev 模式 (**ng build**)
    - src/environments/environment.ts
  - 建置專案的過程中，如果是 prod 模式 (**ng build -prod**)
    - src/environments/environment.**prod**.ts
  - 你也可以自行定義不同的建置模式 (**ng build --env=NAME**)
    - src/environments/environment.**NAME**.ts
- 開發伺服器
  - 用法與 **ng build** 完全一樣
  - **ng serve**
  - **ng serve -prod**

    只要使用 **-prod** 參數，預設會把所有 js 檔案合併 (Bundling)

使用 Angular-CLI 執行測試

- 執行單元測試
  - **ng test**
  - 單元測試會在 ng build 執行完成後，透過 Karma 不斷執行
  - Karma 會自動偵測檔案變更，只要有變更就會自動在背景執行單元測試
  - 你可以執行 **ng test --watch=false** 只執行一次單元測試
  - 也可以執行 **ng test --build=false** 避免執行 **ng build** 建置動作 (當你在執行 **ng serve** 時，專案其實已經都編譯過了，這時就可以用)
  - 在 Windows 平台執行單元測試的注意事項
    - 建議在兩個命令提示字元視窗分別執行以下兩個命令
      ng serve
      ng test --build=false
    - Running `ng test` in windows fails with EMFILE error #977
- 執行 E2E 測試 (End-to-end tests)
  - **ng e2e**
  - 執行之前請先確保 **ng serve** 已在執行中
  - E2E 測試是透過 Protractor 來執行

透過 Angular-CLI 發佈網站到 GitHub.io

- 執行命令
  - `ng github-pages:deploy --message "commit message"`
  - `ng github-pages:deploy --user-page --message "commit message"`
- 首次使用
  - 先到 https://github.com/settings/tokens 建立一個 Token
  - 請勾選 **public_repo** 權限即可 (建立公開的專案)
  - 複製產生的 Token 文字 (一段亂碼字串)
  - 畫面畫上貼上 Token 並輸入你要發佈的 GitHub **帳號**即可！
- 執行過程
  - 建立 GitHub 專案 (repo)，如果專案已存在會中斷執行
  - 將目前最新版本用 Production 模式重建應用程式 (ng build -prod)
  - 建立本地 **gh-pages** 分支 (如果不存在的話)
  - 切換至 **gh-pages** 分支並且建立新的 commit
  - 編輯首頁 **index.html** 的 **<base>** 標籤路徑以正確支援 GitHub Pages
  - 推送 **gh-pages** 分支到 GitHub
  - 回到原本 Git 工作目錄所在的 HEAD 版本

關於 Angular-CLI 注意事項

- 驗證專案語法與格式是否符合標準
  - 執行 `ng lint` 即可進行自動化專案驗證
  - 該命令會在背景執行 `tslint` 命令，會參考 `tslint.json` 定義檔
  - 此命令過成其實只是去執行 `package.json` 裡的 `lint` 命令而已
因此你執行 `npm run lint` 也是完全相同的意思！

- 目前已知的 Angular-CLI 問題
  - 目前產生的專案骨架皆以 TypeScript 為主要語言
  - 在 Windows 平台執行 `ng build` 或 `ng serve` 需要系統管理員權限
    - 不用系統管理員權限執行也可以，只是效能不太好
  - 每次執行 `ng new` 的時間非常久，因為有非常多相依套件需安裝
  - 執行 `ng serve` 的時候要注意專案必須跑在 Node 4 以上版本

Build your Angular 2 Application

建立 ANGULAR 2 應用程式

從現有的 Angular 2 專案範本做起

- 使用 `Webpack`
  - AngularClass/angular2-webpack-starter
  - angular/angular2-seed
- 使用 `SystemJS`
  - DanWahlin/Angular2-JumpStart
  - DanWahlin/Angular2-BareBones
  - johnpapa/angular2-tour-of-heroes
- 使用 `Gulp`
  - `mgechev/angular2-seed`

使用 Angular CLI 建立專案範本

- 請務必在 NTFS 檔案系統執行相關命令 (Windows)
  - `ng new demo1`
    - 初始化 `Git` 儲存庫
    - 安裝 `npm` 相依套件
  - `cd demo1`
  - `ng serve` (若加上 -prod 同時也會關閉 Live Reload)
    - http://localhost:4200
  - `ng generate component header`
  - `ng generate service search`
  - `ng generate pipe mypipe`

- 產生部署網站所需的檔案
  - `ng build`
  - 複製所有 `dist/` 目錄下的檔案進行佈署即可
  - 或可執行 `ng github-pages:deploy` 命令部署到 `GitHub Pages`

了解專案結構

- 首頁 HTML 與 Angular 2 主程式
  - src/index.html　　預設網站首頁
  - src/style.css　　預設全站共用的 CSS 樣式檔
  - src/main.ts　　預設 Angular 2 啟動器 (主程式)
- 公用檔案資料夾
  - src/assets/　　放置網站的相關資源檔案 (CSS,Image,Fonts,...)
- 根元件
  - src/app/index.ts　　載入根元件的預設檔
  - src/app/**app.module.ts**　　應用程式的 NgModule 定義檔
  - src/app/**app.component.ts**　　根元件主程式
  - src/app/**app.component.html**　　根元件範本檔 (Template)
  - src/app/**app.component.css**　　根元件樣式檔 (CSS)
  - src/app/**app.component.spec.ts**　　根元件單元測試定義檔
  - src/app/shared/index.ts　　根元件的共用服務元件或設定
  - src/environments/environment.ts　　環境變數設定 (production: false)
  - src/environments/environment.prod.ts　　環境變數設定 (production: true)

src/index.html

index.html

```
1   <!doctype html>
2   <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Demo1</title>
6     <base href="/">
7
8     <meta name="viewport" content="width=device-width, initial-scale=1">
9     <link rel="icon" type="image/x-icon" href="favicon.ico">
10
11  </head>
12  <body>
13    <app-root>Loading...</app-root>
14  </body>
15  </html>
```

咦？沒有載入任何 JavaScript 函式庫？

根元件的 directive 宣告

src/main.ts

main.ts

```
1   import './polyfills.ts';
2
3   import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
4   import { enableProdMode } from '@angular/core';
5   import { environment } from './environments/environment';
6   import { AppModule } from './app/';
7
8   if (environment.production) {
9     enableProdMode();
10  }
11
12  platformBrowserDynamic().bootstrapModule(AppModule);
13
```

啟用 Production 模式 (提升執行速度)

設定 AppModule 為啟動模組

src/app/app.module.ts

app.module.ts

```
1   import { BrowserModule } from '@angular/platform-browser';
2   import { NgModule } from '@angular/core';
3   import { FormsModule } from '@angular/forms';
4   import { HttpModule } from '@angular/http';
5
6   import { AppComponent } from './app.component';
7
8   @NgModule({
9     declarations: [
10      AppComponent
11    ],
12    imports: [
13      BrowserModule,
14      FormsModule,
15      HttpModule
16    ],
17    providers: [],
18    bootstrap: [AppComponent]
19  })
20  export class AppModule { }
21
```

宣告跟 View 有關的元件

宣告要匯入此模組的外部模組

宣告要註冊的服務元件

宣告根元件

src/app/app.component.ts

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 = 'app works!';
10
11    changeTitle(title: string) {
12      this.title = title;
13    }
14  }
15
```

指令 (directive) 選擇器

元件網頁範本

元件 CSS 樣式

TypeScript 類別

類別中的屬性 (Property)

類別中的方法 (Method)

認識 Angular 2 元件的命名規則

```
// 命名規則： PascalCase
export class AppComponent {
    // 命名規則： camelCase
    pageTitle : string = "Hello World";

    // 命名規則： 動詞 + 名詞 with camelCase
    printTitle() {
        console.log(this.pageTitle);
    }
}
```

建立子元件 (Child Component)

- 建立子元件
  - 透過 **ng generate component *star*** 建立元件
  - 簡寫指令：**ng g c *star***
  - 會建立 **StarComponent** 元件類別

- 在 app.module.ts 匯入 declarations 宣告
  - `import { StarComponent } from './star/star.component';`

- 在上層元件的範本中使用 Directives 語法
  - `<app-star></app-star>`

資料繫結的四種方法 (Binding syntax)

- 內嵌繫結 (interpolation)
  ```
  {{property}}
  ```
- 屬性繫結 (Property Binding)
  ```
  [property]='statement'
  ```
- 事件繫結 (Event Binding)
  ```
  (event)='someMethod($event)'
  ```
- 雙向繫結 (Two-way Binding)
  ```
  [(ngModel)]='property'
  ```

範本參考變數 (Template reference variables)

- 在範本中任意 HTML 標籤套用 `#name` 語法
  - 會在範本內建立一個名為 `name` 的區域變數
  - 該 `name` 區域變數將只能用於目前元件範本中
  - 該 `name` 區域變數將會儲存該標籤的 DOM 物件
  - 你可以透過「事件繫結」將任意 DOM 物件中的任意屬性傳回元件類別中 (Component class)

- 以下這兩種是完全相等的語法 (使用 `#` 是語法糖)
  - `#name`
  - `ref-name`

三種 Angular 指令 (Directives)

https://github.com/miniasp/ng2demo

- 元件型指令
  - 預設「元件」就是一個含有樣板的指令 (最常見)

- 屬性型指令
  - 這種指令會修改元素的外觀或行為
  - 例如內建的 NgStyle 或 NgClass 指令就可讓你自由的變更樣式

- 結構型指令 (Structure Directives)
  - 這種指令會透過新增和刪除 DOM 元素來改變 DOM 結構
  - 例如內建的 NgIf、NgFor 或 NgSwitch 就可以用來控制 DOM 結構
    - 請注意 `ngSwitch` 前面不要加上 * 星號
    - 請注意 `ngIf` 與 `ngFor` 與 `ngSwitchDefault` 與 `ngSwitchCase` 前面要加上 * 星號

關於 * 與 <template> 語法

- 當用到結構型指令時，以下三種寫法都是完全相等的
  -
```
<hero-detail
    *ngIf="currentHero"
    [hero]="currentHero"></hero-detail>
```
  -
```
<hero-detail
    template="ngIf:currentHero"
    [hero]="currentHero"></hero-detail>
```
  -
```
<template [ngIf]="currentHero">
  <hero-detail [hero]="currentHero"></hero-detail>
</template>
```
- 因此套用 * 星號其實是套用 <template> 標籤的**語法糖**
- 請注意上例中 ngIf 所傳入的 "currentHero" 其實是個字串，只要不是空字串都算 Truthy 值，因此不管語法怎麼寫都不可能發生例外！

Angular 2 元件的輸入輸出

- 傳入屬性
  - `@Input() myProperty;`
  - 在**外層元件**請記得用「屬性繫結」傳入資料

- 傳出事件
  - `@Output() myEvent = new EventEmitter();`
  - `this.myEvent.emit(data);`
  - 在**外層元件**請記得用「事件繫結」來接收傳出的資料
    - 在 Template 中使用 **$event** 代表子元件傳出的資料

指令元件的主要生命週期 Hooks

Hook Method	說明
`ngOnInit`	當 Angular 已經初始化過所有 @Input() 屬性後執行 可實作 **`OnInit`** 介面
`ngOnChanges`	當元件的任意 @Input() 屬性被設定後執行 此方法會得到變更物件的目前值與先前的值 可實作 **`OnChanges`** 介面
`ngDoCheck`	當 Angular 每次執行**變更偵測**時執行 (會影響效能)
`ngOnDestroy`	當 Angular 要摧毀元件時執行 建議在此處取消訂閱觀察者物件或刪除先前註冊過的事件處裡器，以避免記憶體洩漏問題發生！ 可實作 **`OnDestroy`** 介面

使用 Pipes

- Angular 2 內建的 Pipes 元件
  - uppercase, lowercase
  - date
  - number, decimal, percent, currency
    - `{{ product.price | currency:'USD':true:'1.2-2' }}`
    - 必須使用 `ISO 4217 currency code` 標準格式
  - json, slice

- Angular 2 並沒有 FilterPipe 與 OrderByPipe 喔！
  - 在 Angular 1 的年代，這兩個經常被濫用且效能低落
  - 因為由於 JS 沒有「傳值」的特性，導致經常有 Bug 出現

Dependency Injection

ANGULAR 2 相依注入

建立服務元件

- 語法結構

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

@Injectable()
export class HeroService {
  getHeroe() { return "HERO";  }
}
```

- 執行命令
  - ng g s hero

設定注入器 (全站共用的服務元件)

- 編輯 app/app.module.ts 檔案

app.module.ts | search.component.ts | app.component.ts

```
import { SearchService } from './search.service';

@NgModule({
  declarations: [
    AppComponent, HeaderComponent, SearchComponent
  ],
  imports: [
    BrowserModule,
    CommonModule,
    FormsModule
  ],
  providers: [SearchService],
  entryComponents: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule {
```

注入服務元件到目前元件

- 程式語法

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

import { SearchService } from '../search.service';

@Component({
  selector: 'app-header',
  templateUrl: 'header.component.html'
})

export class HeaderComponent {
  constructor(searchSvc: SearchService) {
  }
}
```

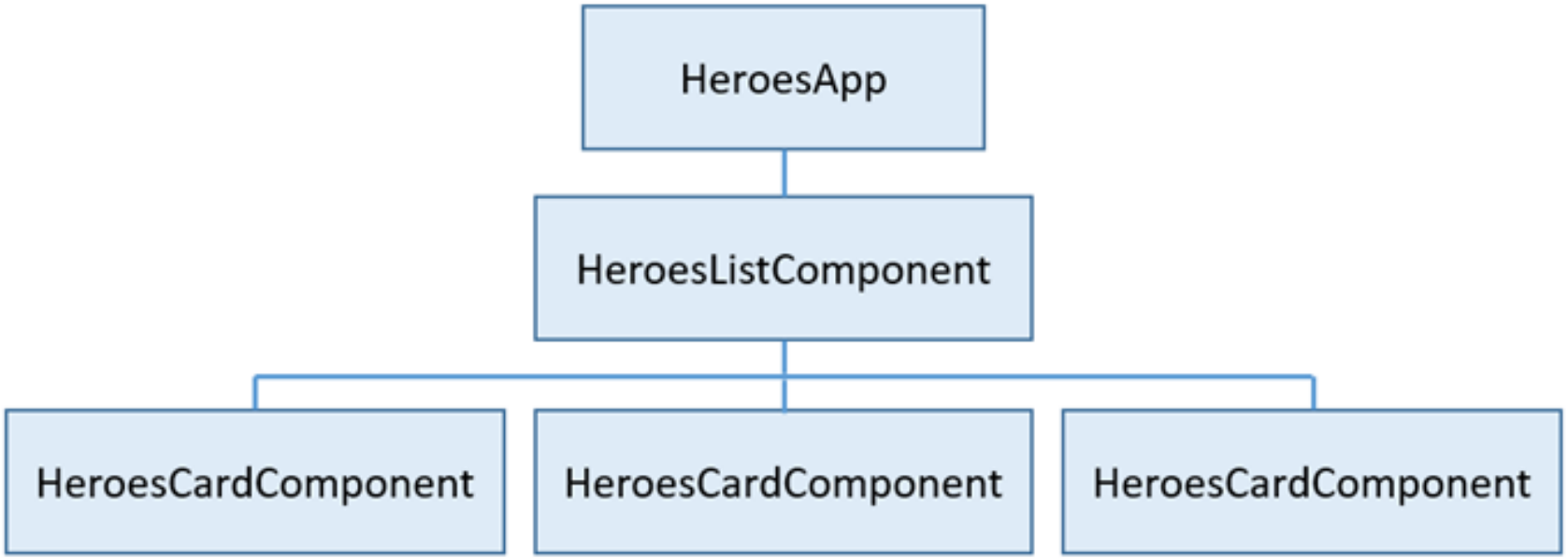
設定注入器 (特定元件下共用的服務元件)

- 編輯特定 *.component.ts 檔案

```
import { SearchService } from './search.service';

@Component({
  selector: 'my-heroes',
  templateUrl: 'some_template_url',
  providers: [SearchService],
  directives: [HeroListComponent]
})
export class HeroesComponent { }
```

注入器的獨體模式 (Singleton)



HTTP CLIENT

使用 HTTP 服務元件

從 AppModule 匯入 HttpModule 模組

app.module.ts

```
1   import { BrowserModule } from '@angular/platform-browser';
2   import { NgModule, ApplicationRef } from '@angular/core';
3   import { CommonModule } from '@angular/common';
4   import { FormsModule } from '@angular/forms';
5   import { AppComponent } from './app.component';
6   import { HttpModule } from '@angular/http';
7
8   @NgModule({
9     declarations: [
10      AppComponent
11    ],
12    imports: [
13      BrowserModule,
14      CommonModule,
15      FormsModule,
16      HttpModule
17    ],
18    providers: [],
19    entryComponents: [AppComponent],
20    bootstrap: [AppComponent]
21  })
22  export class AppModule {
23
24  }
25
```

從元件注入 Http 服務

- 匯入 Http 類別
  - `import { Http } from '@angular/http';`
- 注入 Http 服務
  - `constructor (private http: Http) {}`
- 注意事項
  - 由於原本元件類別的建構式發生變化，因此要注意 *.spec.ts 這個單元測試程式也要跟著修改，否則可能會無法執行！
  - 如果不開發單元測試程式，你也可以考慮直接刪除 *.spec.ts 檔案

發出 HTTP GET 要求與訂閱執行結果

```
import { Headers, RequestOptions, Response } from '@angular/http';

this.http.get('/api/articles.json')
    .subscribe(
        (value: Response) => {
          this.data = value.json();
        },
        (error: any) => {
          this.error = error;
        }
    );
```

Angular2 - set headers for every request - Stack Overflow

發出 HTTP POST 要求與訂閱執行結果

```
import { Headers, RequestOptions, Response } from '@angular/http';

let body = JSON.stringify({ name }); // 先將物件轉 JSON 字串
let headers = new Headers({ 'Content-Type': 'application/json' });
let options = new RequestOptions({ headers: headers });

this.http.post('/api/article', body, options)
    .subscribe(
        (value: Response) => {
          this.data = value.json();
        },
        (error: any) => {
          this.error = error;
        }
    );
```

相關連結

- Angular 2 官網 (官方簡體中文翻譯)
- Angular 2 風格指南 (官方版)
- Angular 2 學習資源 (官方版)
- Angular 2 學習資源 (社群版)
- ng-conf 2016 – YouTube

- Angular 2 Fundamentals | AngularClass (免費 ng2 課程)

- ReactiveX (RxJS on GitHub)
- RxMarbles: Interactive diagrams of Rx Observables

- TypeScript - JavaScript that scales.
- TypeScript Handbook (中文版)

- 前端工程的夢幻逸品：Angular 2 開發框架介紹

聯絡資訊

- The Will Will Web

  記載著 Will 在網路世界的學習心得與技術分享

  - http://blog.miniasp.com/
- Will 保哥的技術交流中心 (臉書粉絲專頁) 

  - http://www.facebook.com/will.fans
- Will 保哥的噗浪

  - http://www.plurk.com/willh/invite
- Will 保哥的推特

  - https://twitter.com/Will_Huang