**Angular Guide**

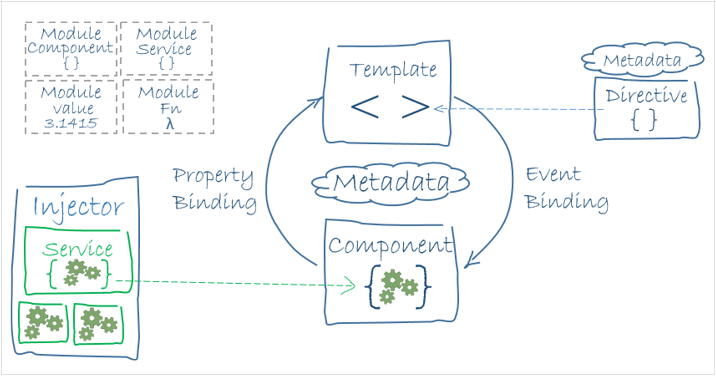
**Architecture Overview**

Angular is a framework for building client applications in HTML and either JavaScript or a language like TypeScript that compiles to JavaScript.

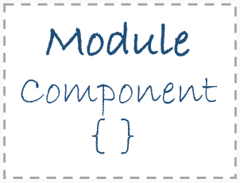
The framework consists of several libraries, some of them core and some optional.

Angular คือเฟรมเวิร์คสำหรับใช้ในการสร้าง client applications ใน HTML ทั้ง JavaScript หรือภาษาเช่น TypeScript ที่คอมไพล์เป็น JavaScript

เฟรมเวิร์คนี้จะประกอบไปด้วยหลาย libraries บางส่วนเป็นไลบรารี่หลักและบางส่วนเป็นไลบรารี่ที่เลือกนำเข้ามาใช้



**Modules**



Angular apps are modular and Angular has its own modularity system called *NgModules*.

NgModules are a big deal. This page introduces modules; the [NgModules](https://angular.io/guide/ngmodules) pages relating to NgModules covers them in detail.

Angular แอพ คือการการแยกส่วนที่เรียกว่า NgModules เป็นเรื่องที่กว้างมาก ในเพจนี้จะแน่ะนำ modules

หน้าเพจ NgModules จะเกี่ยวกับ NgModules และครอบคลุมรายละเอียดดังกล่าว

Every Angular app has at least one NgModule class , [the root module](https://angular.io/guide/bootstrapping), conventionally named AppModule

ทุก Angular แอพจะมีอย่างน้อย 1 NgModules class, root module จะมีชื่อว่า AppModule

* **declarations** - the view classes that belong to this module. Angular has three kinds of view classes: [components](https://angular.io/guide/architecture#components), [directives](https://angular.io/guide/architecture#directives), and [pipes](https://angular.io/guide/pipes).

view classes ที่เป็นของ module นี้ view classes ใน Angular จะมีอยู่ 3 ประเภทคือ [components](https://angular.io/guide/architecture#components), [directives](https://angular.io/guide/architecture#directives), และ [pipes](https://angular.io/guide/pipes)

* **exports** - the subset of declarations that should be visible and usable in the component [templates](https://angular.io/guide/architecture#templates) of other modules.

เป็นสับเซตของ declarations ซึ่งควรจะมองเห็นและใช้ได้ใน component templates ของ module อื่น ๆ

* **imports** - other modules whose exported classes are needed by component templates declared in this module.

โมดูลอื่นที่มีการส่งออก classes และจำเป็นจะต้องใช้โดย component templates จะประกาศใว้ในโมดูลนี้

* **providers** - creators of [services](https://angular.io/guide/architecture#services) that this module contributes to the global collection of services; they become accessible in all parts of the app.

ตัวสร้าง services ที่ module นี้จะสนับสนุนกลุ่มของ services ที่เป็นแบบ global

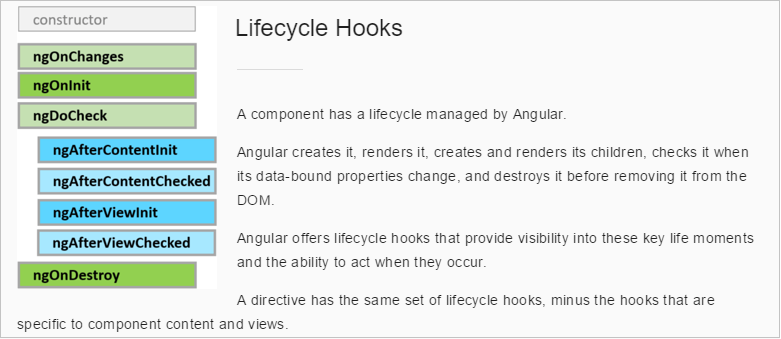
* **bootstrap** - the main application view, called the root component, that hosts all other app views. Only the root module should set this bootstrap property.

เป็นแอพ view หลักที่เรียกว่า root component ซึ่งเป็นเจ้าของแอพ view อื่นๆทั้งหมด เฉพาะ root module ควรจะ set ที่พร็อพเพอร์ตี้ bootstrap นี้

**Component lifecycle hooks overview**

Directive and component instances have a lifecycle as Angular creates, updates, and destroys them. Developers can tap into key moments in that lifecycle by implementing one or more of the lifecycle hook interfaces in the Angular core library.

Each interface has a single hook method whose name is the interface name prefixed with ng. For example, the [OnInit](https://angular.io/api/core/OnInit) interface has a hook method named ngOnInit() that Angular calls shortly after creating the component:



**Lifecycle sequence**

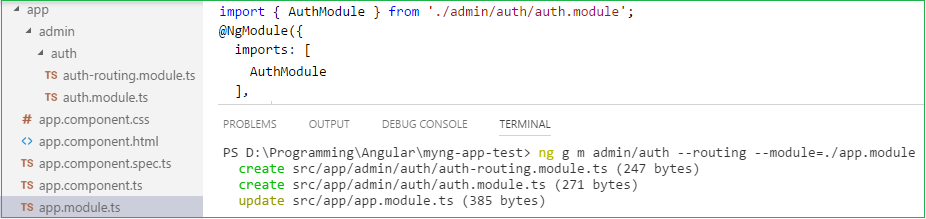
|  |  |
| --- | --- |
| **Hook** | **Purpose and Timing** |
| ngOnChanges() | Respond when Angular (re)sets data-bound input properties. The method receives a [SimpleChanges](https://angular.io/api/core/SimpleChanges) object of current and previous property values.  Called before ngOnInit() and whenever one or more data-bound input properties change. |
|  |
| ngOnInit() | Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties.  Called *once*, after the *first* ngOnChanges(). |
|  |
| ngDoCheck() | Detect and act upon changes that Angular can't or won't detect on its own.  Called during every change detection run, immediately after ngOnChanges() and ngOnInit(). |
|  |
| ngAfterContentInit() | Respond after Angular projects external content into the component's view.  Called *once* after the first ngDoCheck().  *A component-only hook*. |
| ngAfterContentChecked() | Respond after Angular checks the content projected into the component.  Called after the ngAfterContentInit() and every subsequent ngDoCheck().  *A component-only hook*. |
|  |
| ngAfterViewInit() | Respond after Angular initializes the component's views and child views.  Called *once* after the first ngAfterContentChecked().  *A component-only hook*. |
|  |
| ngAfterViewChecked() | Respond after Angular checks the component's views and child views.  Called after the ngAfterViewInit and every subsequent ngAfterContentChecked().  *A component-only hook*. |
|  |
| ngOnDestroy() | Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks.  Called *just before* Angular destroys the directive/component. |
|  |

**การใช้ Angular CLI**

**การสร้าง Component**

* การสร้าง Component แบบที่ 1 (จะสร้างโฟร์เดอร์ให้อัตโนมัติ)
* การสร้าง Component แบบที่ 2 (จะสร้างโฟร์เดอร์ให้อัตโนมัติ) โดยเพิ่มเข้าไปที่ไฟล์ app.module.ts



* **การสร้าง Module และ Route ด้วยคำสั่งเดียว**

คือการสร้าง module และ route จะได้ไฟล์ดังนี้

* auth-routing.module.ts คือการสร้างไฟล์ Route ชื่อ auth-routing และทำการเพิ่มเข้าไปใน auth.module (เพิ่มเข้าไปในไฟล์ auth.module.ts อัตโนมัติ)
* auth.module.ts คือการสร้างไฟล์ module ชื่อ auth.module และจะทำการเพิ่มเข้าไปใน app.module ให้อัตโนมัติด้วยคำสั่ง –module=./app.module นั่นเอง

หมายเหตุ: ถ้ามี module ย่อยเข้าไปข้างในอีกคีย์ต่อท้ายไม่ต้องใส่คำว่า module สมุติว่ามี cart.module ย่อยภายในโฟร์เดอร์อื่นอีกจะใช้คีย์คือ –module=cart

* **การสร้าง Service และเพิ่มเข้าไปยังตำแหน่งของ module ที่ต้องการ**



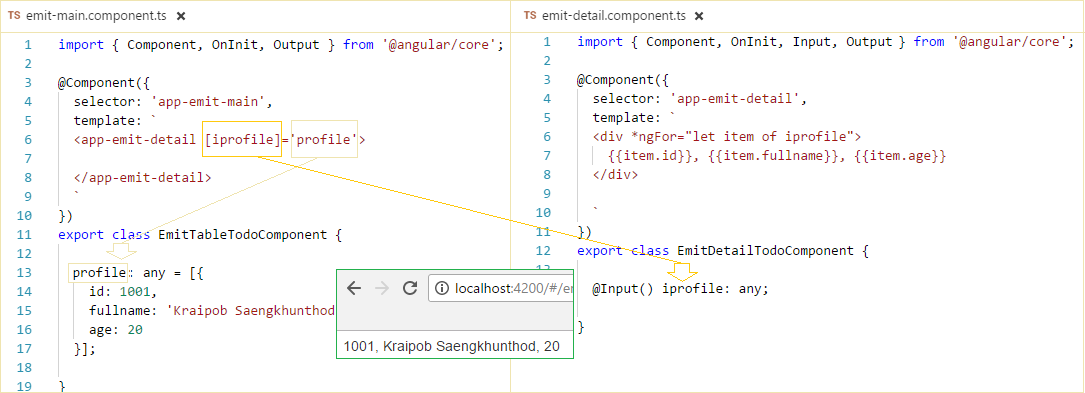
**ตารางคำสั่งต่างๆ**

|  |  |  |
| --- | --- | --- |
| **Scaffold** | **Usage** | **Usage Short** |
| Component | ng generate component my-new-component | ng g c my-new-component |
| Directive | ng generate directive my-new-directive | ng g d my-new-directive |
| Pipe | ng generate pipe my-new-pipe | ng g p my-new-pipe |
| Service | ng generate service my-new-service | ng g s my-new-service |
| Class | ng g class my-new-class | ng g class my-new-class |
| Guard | ng generate guard my-new-guard | ng g g my-new-guard |
| Interface | ng generate interface my-new-interface | ng g i my-new-interface |
| Enum | ng generate enum my-new-enum | ng g e my-new-enum |
| Module | ng g module my-module | ng g m my-module |

**การใช้ Directive และ** **EventEmitt**

* **การใช้ Directive ส่งข้อมูลผ่านไปยัง Coponent อื่น**

Component ที่รับจะต้องมีพร็อพเพอร์ตี้ชนิด @Input() ซึ่งก็คือ iprofile พร็อพเพอร์ตี้ profile ของคอมโพเน้นท์ EmitTableTodoComponent มีชุดข้อมูลชนิดส่งผ่านให้กับ iprofile คอมโพเน้นท์ EmitDetailTodoComponent หรือสามารแก้ใขก่อนแสดงผลเมื่อคอมโพเน้นท์มีการ render ส่วนของ view เรียบร้อยแล้ว เช่น ngOnInit() { this.iprofile[0].fullname = Johnson"; }



|  |  |  |
| --- | --- | --- |
| **No.** | **การค้นคว้า/ทดลอง** | **สถานะ** |
| 1 | การใช้ EventEmitter |  |
|  | * ส่งค่าต่างๆข้าม Component ผ่าน Directive | / |
|  | * การอ้างอิงใช้ค่าร่วมกันระหว่าง Component ผ่าน Directive | / |
|  | * การอ้างอิงใช้ค่าร่วมกันระหว่าง Component และการ Binding Data | / |
|  |  |  |
|  | การสร้างและการใช้งาน Pipe |  |
|  |  |  |