เริ่มต้นใหม่กับ Angular

การมาใหม่ของ Angular มาพร้อมกับโปรแกรมขนาดใหญ่โต จนน่าตกใจ จากเดิม AngularJs มีเพียงไฟล์ angular.min.js ตัวเดียว ของใหม่เริ่มต้นก็ต้องใช้ไฟล์จำนวนมาก ได้แก่โมดูลจำนวนมาก ที่จำเป็น ที่ใช้สร้างเว็บใช้งานขนาดใหญ่ และโมดูลที่เกินความจำเป็น โดยเฉพาะหากต้องการสร้างเว็บใช้งานขนาดเล็ก ซึ่งต่อไปเมื่อสร้างเสร็จรอการใช้งาน ก็จะเหลือแต่โมดูลที่จำเป็นต้องใช้งาน ในจำนวนไฟล์ไม่มากเหมือนตอนแรกที่ใช้สร้างเว็บ ซึ่งแน่นอนว่า จากไฟล์ประมาณ 250 MB จะเหลือประมาณ 250 KB ลดลงไปประมาณ 1000 เท่า เลยทีเดียว และส่วนที่เป็นไฟล์ที่สร้างเพื่อใช้งานเสร็จแล้วนี้ เป็นไฟล์ที่อ่านแทบไม่รู้เรื่องเลย เพราะผ่านกระบวนการ ลดช่องว่าง ปรับเปลี่ยนตัวแปร ดัดที่ไม่ใช้งาน และรวมหลายไฟล์เป็นไฟล์เดียวกัน [1]

ติดตั้ง Angular

Angular ใช้ คำสั่ง CLI (Command Line Interface) ในการสร้างแอปพลิเคชัน และสร้างส่วนประกอบต่าง ๆ ในการพัฒนานาแอปพลิเคชัน แต่ก่อนที่จะติดตั้ง Angular จะต้องติดตั้ง NodeJS ซึ่งเป็นเซิร์บเวอร์ที่ทำงานด้วยภาษา JavaScript ตัวโปรแกรมติดตั้งหาได้จากเว็บไซต์ https://nodejs.org/en/

หลังจากติดตั้ง NodeJs แล้วแต่ใช้คอมมานไล์ สำหรับระบบ Windows ให้เปิด Command Prompt และตอนนี้ต้องให้คอมพิวเตอร์ติดต่อกับอินเทอร์เน็ตได้ด้วย แล้วพิมพ์:

```
npm install -g @angular/cli
```

ต่อมาทดสอบรุ่น (version)

```
ng version
```

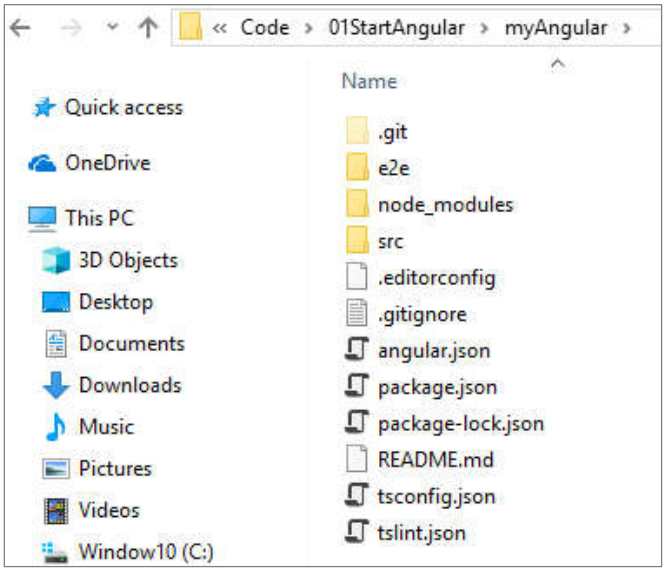
แค่นี้ก็ถือว่าการลงโปรแกรมสมบูรณ์แล้ว

เริ่มสร้างเว็บแอปพลิเคชัน

ก่อนการสร้างเว็บแอปฯ ด้วย Angular จะต้องมีพื้นที่มากพอ ซึ่งไฟล์ทั้งหมดก็ใช้ประมาณ 250 MB การติดตั้งใช้ผ่าน CLI และต้องต่อเชื่อมอินเทอร์เน็ตด้วย มาเริ่มต้นสร้างกันตามลำดับดังนี้

1. `> ng new myAngular`

เมื่อสร้างก็ใช้เวลาไม่มาก สักประมาณ 2 นาที



รูป 1 โครงสร้างไฟล์เริ่มต้นสร้าง Angular

2. ทดสอบการทำงานผ่านเว็บเซิร์บเวอร์ที่มีในตัวอยู่แล้ว ซึ่งจะเปิดเว็บไซต์เริ่มต้นผ่าน localhost:4200

```
>cd myAngular
>ng serve --open
```

นอกจากการแอปพลิเคชันดังตัวอย่างที่ผ่านมายังสามารถใช้ การสร้างในโหมด --strict ได้ด้วย (ใช้กับรุ่นที่ 10 เป็นต้นไป) ด้วยการเขียนเพิ่ม เช่น

```
ng new myAngular --strict
```

การใช้โหมดนี้จะช่วยให้การดูแลรักษาระบบได้ดีขึ้นเนื่องด้วย การแก้ไขความผิดพลาด (bug) ได้ง่ายขึ้น จากการเข้มงวดด้านไวยากรณ์ และข้อกำหนด เช่น การกำหนดชนิดตัวแปรต้องมีแน่ชันเสมอ ดังนั้นจึงไม่สามารถประกาศตัวแปรเป็นชนิด any ได้ และลดจำนวนขนาดแอปพลิเคชันได้ลงขณะเริ่มสร้างแอปพลิเคชัน

หากแอปพลิเคชันเดิมไม่ได้อยู่ในโหมดนี้ ก็สามารถกำหนดเพิ่มเติมไปได้ ดังเขียน:

```
ng generate application [project-name] --strict
```

คอมโพเน้นท์ (Component)

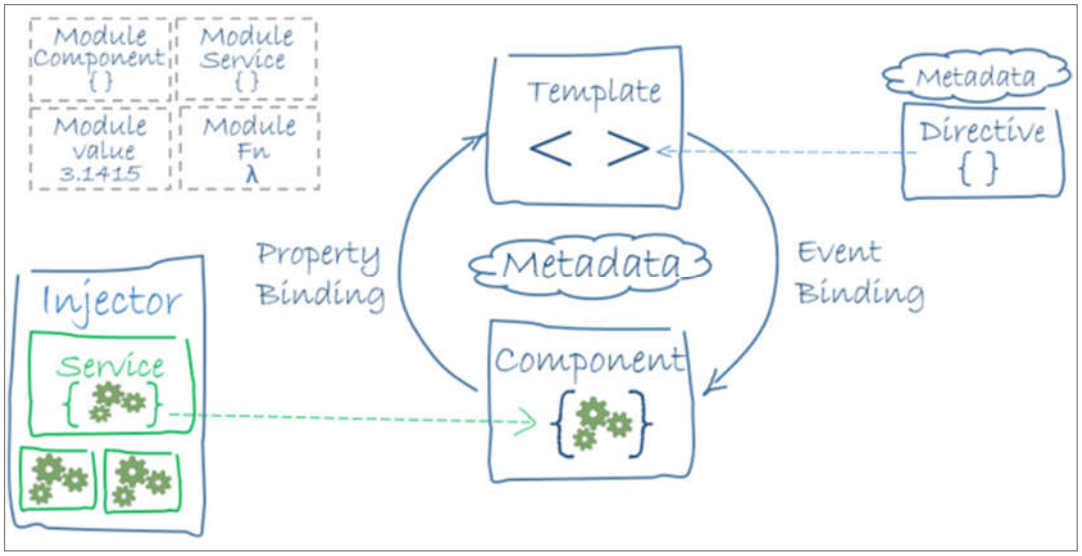
สิ่งสำคัญที่ Angular ใส่เข้ามาใหม่ คือแนวคิด คอมโพเน้นท์ หน้าเว็บหนึ่งคือ คอมโพเน้นท์หนึ่ง คอมโพเน้นท์จะกำหนดหน้าเว็บ หรือวิว (View) ซึ่งจะเรียกใช้งานบริการอื่น ๆ ได้ การเรียกใช้งานผ่านวิธีการที่เรียกกันว่า เดคคอเรเตอร์ (Decorator) เป็นรูปแบบการออกแบบหนึ่งเพื่อเสริมคุณบัติให้หน้าเว็บหนึ่งจะต้องใช้คุณสมบัติอะไรได้บ้าง

ในโครงสร้างของไพล์ที่สร้างมาให้ ให้เปิดที่ myAngular/src/app จะพบไฟล์สำคัญ(ที่ตกแต่งเพิ่ม)คือ:

1. app.component.css เป็นไฟล์ CSS ใช้เพื่อจัดรูปแบบเอกสาร HTML
2. app.component.html เป็นไฟล์หน้าเว็บเริ่มต้น
3. app.component.ts เป็นไฟล์ภาษา TypeScript อยู่ในรูปแบบคลาส ใช้ดำเนินการคอมโพเน้นท์
4. app.module.ts เป็นไฟล์จัดการใช้งานโมดูล ที่จะต้องใช้ในคอมโพเน้นท์ต่างๆ ของเว็บแอปฯ

ดังนี้แล้ว โครงสร้างระบบ หรือสถาปัตยกรรมของ Angular ประกอบด้วยส่วนคอมโพเน้นท์เป็นแกนกลางหลัก ที่เรียกใช้งานงานบริการ โดยมีโมดูลซึ่งเป็นนำไลบรารี่โมดูล (Library module) ไฟล์ JavaScript หรือโมดูลต่าง ๆ เข้าระบบ ในความหมาย JavaScript หนึ่งโมดูลเป็นหนึ่งไฟล์ แต่โมดูลใน Angular ถือเป็นไลบรารีไฟล์ของ JavaScript หลาย ๆ ไฟล์

คอมโพเน้นท์ ทำงานร่วมกับ เทมเพลท (Template) โดยข้อมูลที่กำหนดไว้ในคอมโพเน้นท์ หรือเรียกว่าเมตาดาต้า (Metadata) นำมาอ้างอิงในส่วนแสดงผล ที่อยู่ในรูปแบบไฟล์ HTML และไฟล์ CSS ภายในหน้าแสดงผลนี้ เรียกข้อมูล โดยใช้คำสั่งฝัง (Directive) หรือใช้การผูกข้อมูล (Data/Property binding) หรือข้อมูลมีลักษณะไดนามิกได้ด้วยการใช้ผ่านการผูกกับอีเว้นท์ (Event Binding)



รูป 2 สถาปัยกรรมเว็บแอปฯ ของ Angular

ที่มา : https://angular.io/guide/architecture (12 ธันวาคม 2561)

สำรวจไฟล์คอมโพเน้นท์

คอมโพเน้นท์ส่วนส่วนประกอบหลักของ Angular ที่ใช้ควบคุมการแสดงผล แทนการใช้คอนโทรเลอร์ (Controller) เมื่อเทียบกับ AngularJs ไฟล์ประเภท component.ts จึงนิยามข้อมูลที่ต้องการใช้ในการแสดงผล ซึ่งประกอบได้ด้วย ไฟล์ HTML และไฟล์ CSS โดยชื่ออีลีเม้นท์ (Element) ที่เป็นที่ส่วน HTML จะไปแสดงผล ลักษณะการกำหนดนี้ถือเป็นเมตาดาต้าของคอมโพเน้นท์ และใช้เป็นส่วนเพิ่มไปยังคลาส เรียกส่วนเพิ่มนี้ว่า เดคคอเรเตอร์ กำกับด้วยคีย์สำคัญ @Component

จากไฟล์ต่อไปนี้ (app.component.ts) เป็นคอมโพเน้นท์ชื่อ AppComponent กำหนดเป็นคลาสหนึ่ง การสร้างคลาสประเภทคอมโพเน้นท์ จะต้องประกาศ เดคคอเรเตอร์ ว่า @Component และกำหนดเมตาดาต้า ในตัวอย่างนี้กำหนดส่วนที่แสดงผลในอีลีเม้นท์ <app-root> ซึ่งอยู่ในไฟล์หลัก (src/index.html) และนำไฟล์ app/app.component.html ไปใช้ในอิลีเม้นท์ <app-root> พร้อมกับควบคุมรูปแบบการแสดงผลด้วยไฟล์ app/app.component.css

Code 1. src/app/app.component.ts

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

@Component({
```

```
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'myAngular';
}
```

คลาสนี้จะใช้งานได้จะต้องประกาศคำสำคัญคือ export เพื่อให้นำไปใช้ได้ในโมดูลนี้ การนำคอมโพเน้นท์ไปใช้จะใช้คำสำคัญว่า import ในตัวอย่างนี้ใช้คอมเน้นท์ Component ซึ่งอยู่ในไลบรารี @angular/core นอกจากนี้คลาส AppComponent ประกาศตัวแปร title ซึ่งจะกลายเป็นตัวแปรของออบเจ็กต์ ประจำคลาสนี้

โมดูล

ลักษณะการออกแบบ Angular นอกจากใช้คอมโพเน้นท์เป็นแกนกลางของระบบแล้ว ยังมีส่วนประกอบภายนอกที่คอยเรียกใช้งานไลบรารีในคอมโพเน้นท์ เรียกส่วนนี้ว่าโมดูล โมดูลประกอบด้วยคอมโพเน้นท์ และการเรียกใช้งานโมดูลอื่น ๆ ระบบเว็บแอปฯ หนึ่ง ๆ ของ Angular จะมีอย่างน้อยหนึ่งโมดูล ซึ่งเรียกว่า รูทโมดูล (Root module) หรือโมดูลหลัก และเป็นธรรมเนียมที่จะใช้ชื่อว่า AppModule โดยมูลหลักนี้ เป็นทำงานเป็นที่แรก ซึ่งจะเรียกคอมโพเน้นท์หลักให้ทำงานอีกทีผ่าน การประกาศ @NgModule ซึ่งเป็นเดคอเรเตอร์ประเภทโมดูล ของคลาส AppModule

เดคอเรเตอร์ นี้จะประกาศเมตาดาต้าที่จะต้องนำไปใช้งานในระดับโมดูล เมตาดาต้า มีหลายอย่าง ที่น่าสังเกตว่าข้อมูลใดเป็นอาร์เรย์ จะเขียนในรูป พหูพจน์ (Plural) โดยมากจะใช้อักษร s ต่อท้าย ดังประกาศใช้เช่น :

- declarations เป็นการประกาศคอมโพเน้นท์ ซึ่งอาจจะมี directive, pip ต่าง ๆ นอกเหนือจากคอมโพเน้นท์ ให้อยู่ในโมดูลนี้
- imports เป็นการนำเข้าโมดูลอื่น ๆ ที่จะใช้ในโมดูลนี้
- providers เป็นการประกาศการสร้างงานบริการ (service) ที่โมดูลนี้มีให้ใช้
- boostrap เป็นประกาศคอมโพเน้นท์การประกาศหลัก ที่จะเริ่มทำงาน คอมโพเน้นท์หลัก (root) และมักใช้ชื่อว่า AppComponent

Code 2. src/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

สำรวจไฟล์โมดูล

ต่อไปก็มาดู รายละเอียดไฟล์ที่เพิ่งสร้าง โดยที่ยังไม่เขียนอะไรเพิ่มลงไป ในรายการบรรทัดแรก ๆ จะเป็นการนำเข้าไลบรารี ที่โมดูลจะเรียกใช้งาน โดยรายไลบรารีแรก เป็น ไลบรารี core เพื่อใช้งานโมดูล NgModule ไลบรารี platform-browser เพื่อใช้งานกับเบราเซอร์ และไลบรารี app.component เป็นไลบรารีที่เพิ่มสร้างขึ้นเอง

ในขณะสร้างเริ่มต้น ยังไม่มีงานบริการอะไร ทำให้ providers ยังเป็นอาร์เรย์ว่างอยู่ สำหรับการส่งต่อไปยังการใช้งานในโมดูลอื่น ด้วยคำสำคัญ export โมดูลนี้ซึ่งเป็นโมดูลหลัก หากว่าโมดูลอื่น ต้องการใช้งานก็จะนำโมดูลนี้ไปใช้งานต่อได้ แต่ก็ไม่จำเป็นเพราะเราสร้างโมดูลนี้เพื่อการใช้งานในตัวเองเท่านั้น

Code 3. src/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

เดคคอเรเตอร์

แนวการเขียนโปรแกรมที่ต้องการเสริมคุณบัติของสิ่งที่ต้องการขยาย โดยการประกาศแจ้ง (annotating) ด้วยเครื่องหมาย @ นำหน้า แยกได้ได้หลายประเภท ซึ่งทั้งหมดถือเป็นเขียนโปรแกรมแบบ เมตา-ดาต้า (Metadata-programming) หรือการให้ความหมายใหม่ตามประกาศ ประเภทหลักคือ

- ประกาศบนคลาส (class decorators)
- ประกาศบนฟังก์ชัน/เมธอด (method decorators)
- ประกาศภายในคลาส (property decorators)
- ประกาศเป็นตัวแปรของฟังก์ชัน (parameter decorator)

ในเดคคอเรเตอร์ประเภทต่างๆ มีตัวหนึ่งที่พบบ่อยคือ @Component({ }) เป็นการประกาศก่อนประกาศคลาส ภายใน @Component นี้ก็จะมีจะให้ความหมายเสริม ในรูปออบเจ็กต์ อันประกอบด้วยสมาชิก selector, templateUrl, และ styleUrls ตามข้อกำหนดมีได้ใน @Component นี้

ต่อไปจะยกตัวอย่างการสร้าง เดคคอเรเตอร์ Directive หรือเป็นการสร้างส่วนเสริมคำสั่งที่ใช้ กับ HTML สมมุติต้องการให้ชื่อว่า item จะใช้คำสั่งใน CLI จะทำให้เกิดไฟล์ใหม่คือ item.directive.ts และเพิ่มส่วน item นี้ลงในไฟล์ app.module.ts ต่อไปนี้ (ตอนนี้อยู่ในไดเรกทรอรี่ angular) :

```
myAngular> ng generate directive item
```

หรือจะสร้างไฟล์ขึ้นมาเองก็ได้ ในชื่อ item.directive.ts

@Directive

คำสั่งที่ฝังใน HTML ที่สร้างขึ้นเอง เรียกกันว่า ไดเร็กทีฟ(directive) เช่น ต้องการสร้าง คำสั่งฝังที่จะใช้เป็นหนึ่งของอิลีเม้นท์ HTML ว่า appItem จะต้องสร้างไฟล์ คำสั่งฝังนี้ก่อน และดัดแปลงเพิ่มการนำไปใช้กับ CSS ดังตัวอย่างต่อไปนี้

Code 4. src/app/item.directive.ts

```
import { Directive , ElementRef } from '@angular/core';

@Directive({
  selector: '[appItem]'
})
export class ItemDirective {
  constructor(el:ElementRef) {
      el.nativeElement.style.backgroundColor = '#ddd';
  }
}
```

การเพิ่ม ElementRef มีนี้ถือเป็นการใช้งานกับ อิลีเม้นท์ของ HTML และนำไปประยุกต์กับ CSS ได้ ในที่นี้กำหนดให้สีพื้นหลังเป็นสีเทาอ่อน (#ddd)

เมื่อสร้างไฟล์สำหรับคำสั่งฝังแล้ว จะต้องประกาศในโมดูล ให้รู้ที่อยู่ว่าอยู่ที่ไฟล์ใด ซึ่งใช้คำสั่ง import ในชื่อคลาสและชื่อไฟล์

นอกจากนี้ยังต้องประกาศในส่วน declarations ด้วยว่าจะใช้คำสั่งฝังนี้ในโมดูล ซึ่งเพิ่มเติมจากการประกาศคอมโพเน้นท์ ให้อยู่ในรูปอาร์เรย์ ดังตัวอย่างต่อไปนี้

Code 5. src/app.module.ts (บางส่วน)

```
import { ItemDirective } from './item.directive';
declarations: [
  AppComponent,
  ItemDirective
],
```

ต่อมาก็นำ Directive นี้ไปใช้งาน ด้วยการใส่ appItem ไว้ใน <p> ให้เพิ่มอิลีเม้นท์ <p> ต่อไปนี้ไว้บรรทัดสุดท้ายของไฟล์ app.component.html

Code 6. src/app.component.html (บางส่วน)

```
<p appItem>Hello Directive</p>
```

รูป 3 หน้าเว็บแอปฯ ของ Angular

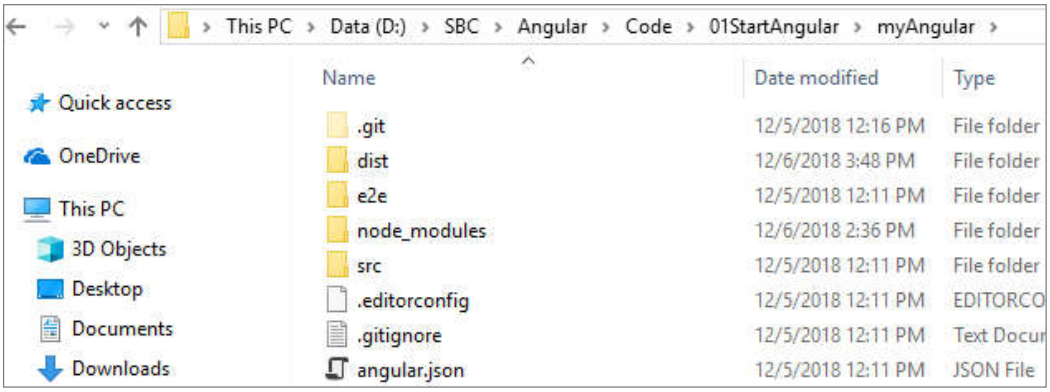
การนำไปใช้งานจริง

ณ ขณะสร้างนี้ เราสร้างขึ้นโดยไม่ได้แต่งเติมอะไร ทำตามค่าปริยายที่ Angular ทำให้เองทั้งหมด ซึ่งเราก็ได้ทดลองเปิดไฟล์ ผ่าน ng serve --open แล้ว ซึ่งมีเพียงหน้าเดียว แต่ประกอบด้วยไฟล์จำนวนมาก

ต่อไปเมื่อนำไปใช้งานจริง (เว็บหน้าเดียวนี้) จะต้องทำการสร้างเป็นไฟล์ ลดขนาด ให้เล็กลง ขั้นตอนนี้เรียกว่า การผลิตงาน (Production) ให้เข้าไปยังตำแหน่ง โฟลเดอร์งานปัจจุบัน เช่น ณ ตอนแรกที่สร้างจะได้โฟลเดอร์ myAngular ให้พิมพ์คำสั่งต่อไปนี้ผ่าน CLI

```
>ng build --prod
```

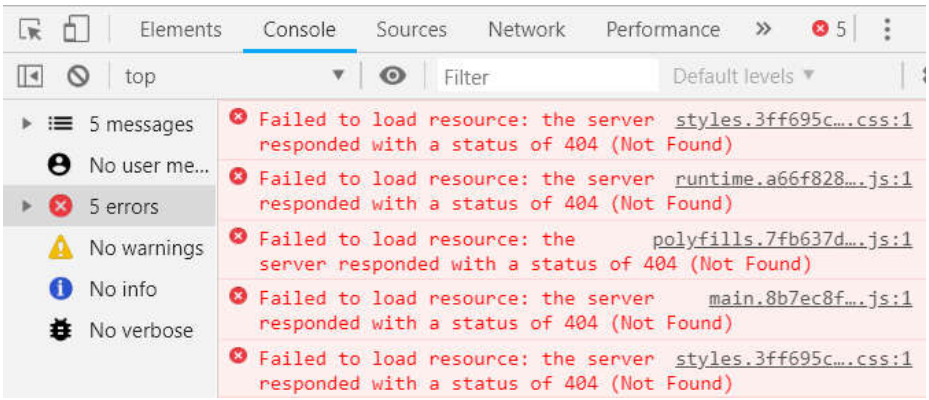
ขั้นตอนนี้ Angular จะสร้างไฟล์ใหม่อยู่ใน โฟลเดอร์ dist ซึ่งย่อมาจากคำว่า distribution แปลว่านำไปแจกจ่ายใช้งานได้



รูป 4 ผลการผลิตงาน จะได้โฟลเดอร์ dist

ไฟล์ต่างๆ ที่อยู่ ในโฟเดอร์ dist ทั้งหมด เมื่อเปิดไฟล์ ดูรายละเอียดข้างในดู จะพบว่า ไฟล์ มีการลดขนาดลงไปมาก จะมีเพียงไฟล์ index.html เท่านั้นที่อ่านเข้าใจได้และมีเพียงไม่กี่บรรทัด ส่วนไฟล์อื่น ๆ ถูกย่อจนอ่านแทบไม่ได้เลย ไฟล์ต่างๆ เหล่านี้ สามารถนำไปใช้งานได้ โดยนำไปเก็บที่เว็บเซร์บเวอร์ได้ โดยการที่ราก (root) ของเว็บ แล้วทดลองเปิดผ่านเบราเซอร์ ถ้าไม่มีอะไรผิดพลาดจะเปิดได้เหมือนกับที่เคยเปิดมาก่อนหน้านี้

แต่อย่างไรก็ตาม ในการทดลองใช้งานนี้เรามักวางไฟล์เหล่านี้ ในตำแหน่ง ที่ไม่ได้อยู่ที่ราก หรือตำแหน่งแรกสุดของเว็บเซิร์บเวอร์ เช่น ถ้าว่าง ใน Appserv จะต้องวางในโฟลเดอร์ www เช่นวางที่ www/myAngular โดยไฟล์ทั้งหมดของภายใน dist ให้อยู่ในโฟลเดอร์ myAngular หรือถ้าใช้โปรแกรม EasyPHP ที่ให้สร้างโฟลเดอร์เก็บงานที่ใดก็ได้โดยไม่จำเป็นต้องอยู่ในส่วนโฟลเดอร์ของโปรแกรม EasyPHP เช่นวางที่ D:/myAngular ดังนั้นการเปิดไฟล์ผ่านเบราเซอร์ เช่น localhost/myAngular หรือ http://127.0.0.1:8080/edsa-myAngular/ เว็บจึงไม่ได้อยู่ที่ตำแหน่งราก ของเว็บเซิร์บเวอร์ เมื่อเปิด ผ่านดูในตำแหน่ง url ข้างต้นจะพบว่า เป็นไฟล์หน้าเปล่า ไม่มีอะไรขึ้นเลย และถ้าใช้ Google Chrome เปิดหน้าต่าง Developer Tools จะพบ การหาไฟล์ที่ต้องใช้งานไม่เจอ นี้เป็นสาเหตุเว็บที่ผลมากจากการผลิตทำงานไม่ได้



รูป 5 หน้าต่าง More Tools / Developer Tools / Console

วิธีการแก้ไขให้กำหนดเส้นทางของไฟล์ใหม่ โดยการเปิด ไฟล์ index.html ซึ่งจะพบ ว่า `<base href="/">` ในบรรทัดที่ 6 ตาม **รูป 6** จะเขียนตำแหน่งไฟล์เริ่มต้นที่รากของเว็บเซิร์บเวอร์ ทำให้เกิดความผิดพลาดในการหาไฟล์ไม่พบ ให้แก้โดยการใส่เส้นทาง เช่น กรณีใช้ http://127.0.0.1:8080/edsa-myAngular/ มีส่วนต่อท้ายเป็น edsa-myAngular เติมต่อมาจากตำแหน่งราก จึงต้องแก้ให้ การอ้างอิงในส่วนเติมจากรากของเว็บเซิร์บเวอร์ด้วย คือต้องใส่เป็น

```
<base href="edsa-myAngular">
```

อีกวิธีหนึ่ง ซึ่งได้ผลเหมือนกันกับการเขียนเติมในไฟล์ index.html โดยการแก้ไข ขณะ ที่กระบวนการผลิตงาน จากเดิมใช้เพียงคำสั่ง --prod ต่อท้าย ให้เพิ่มคำสั่งเพิ่มดังนี้

```
>ng build --prod --base-href edsa-myAngular
```

ถึงตอนนี้ หวังว่า รู้สึกว่าเข้าใจขึ้นมาบ้าง และเห็นประโยชน์อันมาก ของ Angular ความน่าใช้งาน น่าเรียนรู้ นี้จะเป็นกระตุ้นให้ศึกษาการใช้งาน Angular มากขึ้น แล้วพบกันใหม่ในบทต่อไป

```
1   <!doctype html>
2   <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>MyAngular</title>
6     <base href="/">
7     <meta name="viewport" content="width=device
8     <link rel="icon" type="image/x-icon" href="
9   <link rel="stylesheet" href="styles.3ff695c0
10  <body>
11    <app-root></app-root>
12  <script type="text/javascript" src="runtime.
13  </html>
```

รูป 6 index.html

สรุป

ในบทนี้ ต้องการทำให้เห็นว่า Angular น่าใช้งานเพียงใด โดยให้สร้างเป็น แอปพลิเคชั่นอย่างง่าย โดยไม่ต้องเขียนอะไรเพิ่มเติมมากมายนัก ในการสร้างแอปฯ ให้ทำความรู้จักโครงสร้างโดยรวม ของโมดูล โดยเฉพาะคอมโพเน้นท์ ที่จะเป็นหน้าเว็บที่แสดงผล โดยแนวคิดใหม่ที่ให้คอมโพเน้นท์หนึ่ง เป็นหนึ่งหน้าเว็บ ที่ประกอบด้วยไฟล์ HTML และไฟล์ TS เป็นหลัก โดย TS ทำหน้าที่ควบคุมการแสดงผล และข้อมูลที่ต้องการใช้ใน HTML นอกจากนี้ ยังให้ทดลองคอมไฟล์ไฟล์ทั้งหมด เพื่อการนำไปใช้งาน จากตัวอย่างจะพบแล้วว่า ไฟล์มีขนาดเล็กลงมาก ซึ่งเป็นเสน่ห์ อย่างหนึ่งที่น่าสนใจ Angular ขึ้นมาก เราจะได้ศึกษาความน่าสนใจอื่น ๆ อีก ที่ทำให้ Angular เหมาะสมกับแอปฯ ขนาดใหญ่ ๆ และเป็นจุดสำคัญที่ต้องหนมาใช้ Angular แทน AngualrJs

อ้างอิง/อ่านเพิ่มเติม

[1] Medium : Coinmonks. https://medium.com/coinmonks/how-to-deploy-an-angular-app-8db1af39f8c1. (6 Nov., 2018).

[2] Medium : List of all @Decorator available in a Angular. (17 Apr. 2020). https://medium.com/@madhavmahesh/list-of-all-decorators-available-in-angular-71bdf4ad6976

[3] Typescript: Decorators. (17 Apr. 2020). https://www.typescriptlang.org/docs/handbook/decorators.html

[4] Todd Motto. A deep dive on Angular. (17 Apr. 2020). https://ultimatecourses.com/blog/angular-decorators