

แองกูลาร์ แมทเทอร์เรียลดีไซน์

แองกูลาร์ แมทเทอร์เรียลดีไซน์ (Material Design) เป็นคอมโพเนนต์ที่ได้รับการออกแบบมาเพื่อใช้งานโดยเฉพาะกับแองกูลาร์ การใช้คำว่า แมทเทอร์เรียล ซึ่งแปลว่าวัสดุ เป็นการใช้อุปมาอุปมัย แทนสิ่งที่ออกแบบให้ผู้ใช้งานสัมพันธ์กับระบบ การใช้ แมทเทอร์เรียลดีไซน์ ไม่ได้สร้างมาเฉพาะกับเว็บไซต์เท่านั้น แต่ยังสามารถใช้ได้กับระบบอื่น ๆ เช่น ระบบแอนดรอยด์ ไอโอเอส ในบทนี้เราจะมีเริ่มทำความเข้าใจกับแมทเทอร์เรียล ในประเด็นดังนี้

- การใช้ สกีแมติกส์ (Schematics) ในการสร้างเว็บตั้งต้น
- การประยุกต์ระบบนำทาง
- การใช้งานไอคอน (Icon) รูปภาพ
- การใช้งานไดอะล็อก (Dialog)
- การสร้างฟอร์ม
- การใช้งานตาราง

เราจะเริ่มจากการสร้างคอมโพเนนต์สำเร็จรูป เป็นรูปแบบที่เรียกว่า สกีแมติกส์ ซึ่งมีด้วยกันหลายรูปแบบซึ่งสร้างเป็นออกมาเป็นคอมโพเนนต์ตามความต้องการใช้งาน สกีแมติกส์ แรกเราเริ่มจาก คอมโพเนนต์รายการนำทาง (Navigation) เป็นโครงสร้างพื้นฐาน ที่จะนำไปสู่สกีแมติกส์อื่น ๆ และการใส่แมทเทอร์เรียลดีไซน์ รวมถึงการปรับแต่งแมทเทอร์เรียลดีไซน์

เริ่มต้นใช้ แมทเทอร์เรียลดีไซน์ ด้วยสกีแมติกส์นำทาง (Navigation)

1. สร้างโปรเจกต์ใหม่ เลือกแบบมีเส้นทาง

```
ng new myAngular --routing
cd myAngular
```

2. เพิ่ม Material Design สู่โปรเจกต์

```
ng add @angular/material
```

ในขั้นตอนนี้ CLI จะถามหลายคำถามในการกำหนดค่าเริ่มต้น คือ

- ให้เลือก theme เลือกตัวแรกซึ่งเป็น Indigo/Pink ซึ่งจะเลือกตัวอื่นก็ได้
- กำหนดค่า Typography Styles แบบ global หรือไม่ให้ตอบ ใช่ (y)
- กำหนดค่าการเคลื่อนไหว (browser animation) ให้ตอบ ใช่ (y)

3. สร้างคอมโพเนนต์ navigation เป็นเหมือนเมนูนำทาง ซึ่งเป็นคอมโพเนนต์สำเร็จรูปตัวหนึ่งของ Schematics

```
ng generate @angular/material:navigation navigation
```

4. เปิดไฟล์ app.component.html แล้วแทนที่ข้อมูลเดิมทั้งหมดด้วยข้อความต่อไปนี้

```
<app-navigation></app-navigation>
```

5. เปิดการทำงาน

```
ng serve --open
```

6. ผลที่ได้จากเปิดเบราว์เซอร์ <http://localhost:4200> ดังรูปต่อไปนี้ ซึ่งถ้าย่อขนาดความกว้างของจอลง เมนูซ้ายมือจะหายไป แล้วจะกลายเป็นปุ่มเมนูบนทูลบาร์ (tool bar) แทน

Menu	myAngular
Link 1	
Link 2	
Link 3	

สกีแมติกส์นำทาง

รูปแบบของคอมโพเนนต์นี้ประกอบด้วยหลายโมดูลย่อย ซึ่งถ้าเปิดดูในโมดูลจะพบว่ามี การนำเข้าโมดูลมากมาย แต่โมดูลทำหน้าที่ต่าง ๆ กัน เช่น มีโมดูล LayoutModule ช่วยจัดวางผังเว็บให้ตอบสนองกับความกว้างของหน้าจอให้เหมาะสม มีตัวสังเกตการณ์ BreakpointObserver เพื่อสืบค้นผ่าน media query แล้วประเมินผลให้แสดงผลให้เหมาะสมกับขนาดอุปกรณ์ที่แสดงผล

ตาราง 1 โมดูลใน Navigation Schematic

โมดูล	ชื่ออ้างอิง	ทำหน้าที่
LayoutModule	BreakpointObserver, Breakpoints	ตอบสนองกับ UI ตามขนาดหน้าจอ ใช้อ้างในคลาส NavigatorComponent
MatToolbarModule	<mat-toolbar>	เป็นส่วนหัวรายการ ใส่ชื่อรายการ และปุ่มคำสั่งได้
MatSidenavModule	<mat-sidenav>	แสดงเป็นเมนูด้านข้าง ซึ่งเลื่อนเข้าออกได้
MatIconModule	<mat-icon>	เป็นไปไอคอน ซึ่งแสดงเป็นรูปภาพต่างๆ
MatListModule	<mat-list-item>	แสดงเป็นรายการเมื่อนำทาง
MatButtonModule	<mat-icon-button>	ปุ่มคำสั่ง

หากเปิดคลาสหลักของคอมโพเนนต์นี้จะพบงานบริการ isHandset\$ ซึ่งใช้ตรวจสอบขนาดการวาง (layout) ของหน้าจอ ซึ่งการตรวจสอบนี้ใช้ breakpointObserver และ Breakpoints ซึ่งทั้งสองตัวนี้อยู่ในแพ็คเกจ Layout ดังพิจารณาจากตัวอย่างต่อไปนี้

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

```
export class NavigationComponent {
```

```

    isHandset$: Observable<boolean> = this.breakpointObserver
        .observe(Breakpoints.Handset)
        .pipe(map(result =>
            result.matches),
            shareReplay()
        );
    constructor(private breakpointObserver: BreakpointObserver) {}
}

```

จากตัวอย่างนี้สร้างงานบริการ isHandset\$ ซึ่งจะรับบริการผ่านการผูกข้อมูลกับหน้า HTML โดยใช้คีย์เวิร์ด async ผลของการผูกข้อมูลจะได้ค่าจริงหรือค่าเท็จ ซึ่งเป็นผลมาจากตรวจสอบขนาดหน้าจอที่มีค่าปรีายของค่า breakpoint ที่มีค่าตรงกับค่าปรีาย (handset) หรือไม่ ค่า handset คือค่าที่เหมาะสมกับอุปกรณ์โทรศัพท์¹

โครงสร้างของ Sidenav

โครงสร้างพื้นฐานของรายการนำทางนี้ประกอบด้วยสองส่วนคือ <mat-sidenav> ทำหน้าที่เป็นเมนูนำทางด้านซ้าย และ <mat-sidenav-content> ทำหน้าที่เป็นส่วนแสดงเนื้อหา และทั้งสองส่วนนี้บรรจุอยู่ใน <mat-sidenav-container> ดังแสดงได้ตามตัวอย่างต่อไปนี้

Code 2.

```

<mat-sidenav-container class="example-container" *ngIf="shouldRun">
  <mat-sidenav mode="side" opened>Sidenav content</mat-sidenav>
  <mat-sidenav-content>Main content</mat-sidenav-content>
</mat-sidenav-container>

```

จากตัวอย่างนี้จะเห็นว่า ซึ่งมีเพียง <mat-sidenav> ตัวเดียว จึงเป็นค่าปรีายที่อยู่ทางซ้ายมือ ค่าปรีายคือ position="start" แต่ถ้าเราระบุตำแหน่งเป็น position="end" จะทำให้ <mat-sidenav> แสดงผลด้านขวามือ แต่ <mat-sidenav> ตัวแรกต้องมีตำแหน่งซ้ายมือเสมอ (ถ้ามีสองตัว) ตัวอย่างต่อไปนี้เป็นตัวอย่างที่ผิด เพราะมีตัวแรกของ <mat-sidenav> มีตำแหน่งเป็น start และตัวที่สองก็เป็น start ด้วย

Code 3. ตัวอย่างที่ผิด

```

<mat-sidenav-container>
  <mat-sidenav>Start</mat-sidenav>
  <mat-sidenav position="start">Start 2</mat-sidenav>
</mat-sidenav-container>

```

นอกจากนี้ยังกำหนดคุณสมบัติ opened ทำให้มีการแสดงผลเสมอของ <mat-sidenav> แต่ถ้ากำหนดเป็น closed จะเปิดการปิด หรือไม่แสดงผลของ <mat-sidenav>

สำหรับการกำหนดคุณสมบัติ mode เป็น side หมายถึงแสดงผลทางด้านข้าง ยังมี mode เป็น over ซึ่งเป็นค่าปรีายที่จะแสดงผลทับ <mat-sidenav-content> และพื้นหลังเป็นสีเทา การที่จะไม่แสดงผลทับก็สามารถกำหนดได้ คุณสมบัติ hasBackdrop เป็น true หรือ false ไว้ที่ <mat-sidenav-container>

¹ตรวจได้จากข้อกำหนดจากเว็บ <https://material.io/design/layout/responsive-layout-grid.html#breakpoints>

การกำหนดขนาดของ <mat-sidenav> ได้ถูกกำหนดไว้แล้วที่ไฟล์ CSS อย่างไรก็ตามหากข้อความใน <mat-sidenav> มีมากจึงต้องการขนาดที่กว้างที่กำหนด ขนาดของ <mat-sidenav> ก็จะกว้างขึ้นอัตโนมัติ

ปรับแต่งรายการนำทางใหม่

จากหน้านำทางที่ได้สร้าง จะมีเมนูด้านซ้ายเสมอเมื่อจอมีขนาดใหญ่ แต่ก็เสียพื้นที่ด้านซ้ายไป เมนูควรจะอยู่ด้านบน ซึ่งว่างอยู่มากกว่า ดังนั้นเมื่อต้องการปรับแต่งใหม่ตามความต้องการนี้ควรจะแก้ไขใหม่

สิ่งแรกที่ต้องทำคือ ทำให้เมนูซ้ายมือหายไป โดยการลบส่วนเงื่อนไขการเปิดออก (opened) ของ <mat-sidenav> ดังนั้นส่วนเหลือดังตัวอย่างต่อไปนี้

Code 4. src/app/navigation/navigation.component.html

```
<mat-sidenav #drawer class="sidenav"
  [attr.role]="(isHandset$ | async) ? 'dialog' : 'navigation'"
  [mode]="(isHandset$ | async) ? 'over' : 'side'">
```



รูป 1 รายการนำทางที่ต้องการแก้ไข

ต่อมา เปิดปุ่มคำสั่งใน <mat-toolbar> ในตัวอย่างต่อไปนี้ใช้ ปุ่มคำสั่ง 4 ตัวคือ Courses, Users, About Us, และ Signup ทั้งสามปุ่มคำสั่งใช้ mat-button ส่วนปุ่มสุดท้ายเปลี่ยนสีเป็น accent

นอกจากนี้ก่อนสามปุ่มนี้มี justify-content: flex: 1 เพื่อให้ <div> ขยายเต็มพื้นที่ซึ่งทำให้ ด้านทั้งสามปุ่มคำสั่งไปทางขวาสุด และ mat-elevation-z6 เป็นการสร้างเงาของ <mat-toolbar> ตัวเลขสูงขึ้นจะเป็นการเพิ่มระดับเงามากขึ้น

Code 5. src/app/navigation/navigation.component.html

```
<mat-toolbar color="primary" class="mat-elevation-z6">
  <button type="button" aria-label="Toggle sidenav" mat-icon-button
    (click)="drawer.toggle()" *ngIf="isHandset$ | async">
    <mat-icon aria-label="Side nav toggle icon">menu</mat-icon>
  </button>
  <span>myAngular</span>
  <div style='flex:1'></div>
  <a mat-button>Courses</a>
  <a mat-button>Users</a>
  <a mat-button>About Us</a>
  <a mat-raised-button color='accent'>Sign up</a>
</mat-toolbar>
```

สีของ <mat-toolbar> (ตัวบน) ควรเป็นสีเดียวกันหมด จากที่สร้างมาให้มี <mat-toolbar> มาสองตัว แต่มีอีกตัวที่ยังไม่เป็นสี primary คือตัวที่อยู่ใน <mat-sidenav> จึงควรแก้ไขให้สีให้เหมือนกันด้วย แต่เนื่องจากใน ไฟล์ CSS ได้กำหนดให้สีพื้นแบบ inherit จึงควรปิดการทำงานส่วนนี้เสีย

Code 6. src/app/navigation/navigation.component.css

```
/* หรือลบส่วน CSS นี้ออกไปเลย
.sidenav .mat-toolbar {
  background: inherit;
```

```
}
*/
```

แล้วใส่แก้ไขให้สีเหมือนตัวล่าง จึงแก้การใช้สีในแบบ primary ให้เหมือนกันดังนี้

Code 7. src/app/navigation/navigation.component.html

```
<mat-toolbar color="primary">Menu</mat-toolbar>
```

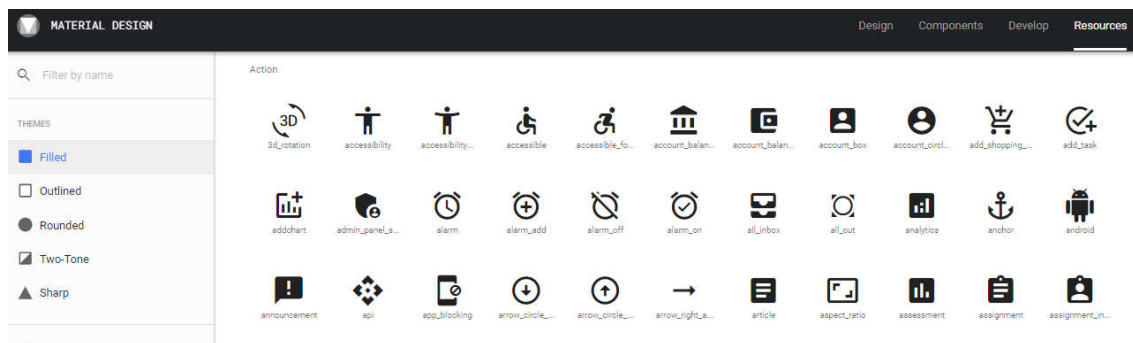
สร้างไอคอนให้กับรายการนำทาง

การสร้างไอคอน (Icon) ซึ่งเป็นภาพขนาดเล็ก และเพิ่มตัวชี้รายการ จะต้องทำเข้าโมดูล MatIconModule แต่ใน Schematic: navigation ได้นำเข้าโมดูลนี้แล้ว ส่วน MatDividerModule ยังไม่มีการนำเข้าจึงต้องเพิ่มนำเข้าโมดูลนี้ และในส่วนอาร์เรย์ imports[] ด้วย

Code 8. src/app.module.ts

```
import { MatDividerModule } from '@angular/material/divider';
```

รูปแบบการใช้ไอคอน ต้องกำหนดใน <mat-icon> ภายในอีลีเมนต์นี้ต้องใส่ชื่อของไอคอน เช่น ชื่อ group จะเป็นรูปคนสองคน ส่วนการเพิ่มคุณสมบัติอื่น เช่น aria-hidden เป็นการบอกว่าโปรแกรมอ่านออกเสียงของเว็บว่าไม่ต้องสนใจ (อ่าน) เพราะบางทีโปรแกรมอ่านออกเสียงจะอ่านไปด้วยเพื่อคนพิการทางสายตา



รูป 2 ไอคอนต่างๆ ใน Material Design²

การใส่สีก็เป็นคุณสมบัติหนึ่งที่ทำให้เหมาะกับแต่ละรูปแบบโทนสีของเว็บ การกำหนดทำผ่านคุณสมบัติ color แล้วเลือกสี เช่น accent, warn, primary

```
<mat-icon aria-hidden="false" color='accent'>group</mat-icon>
```

ดังนั้นแล้ว ไอคอนที่จะใส่ในรายการนำทางในส่วน <mat-nav-list> ในรายการนำทางต่าง ๆ ดังเป็นตัวอย่างที่ได้ปรับปรุงใหม่ดังนี้

Code 9. src/app/navigation/navigation.component.html

```
<mat-nav-list>
  <mat-divider></mat-divider>
```

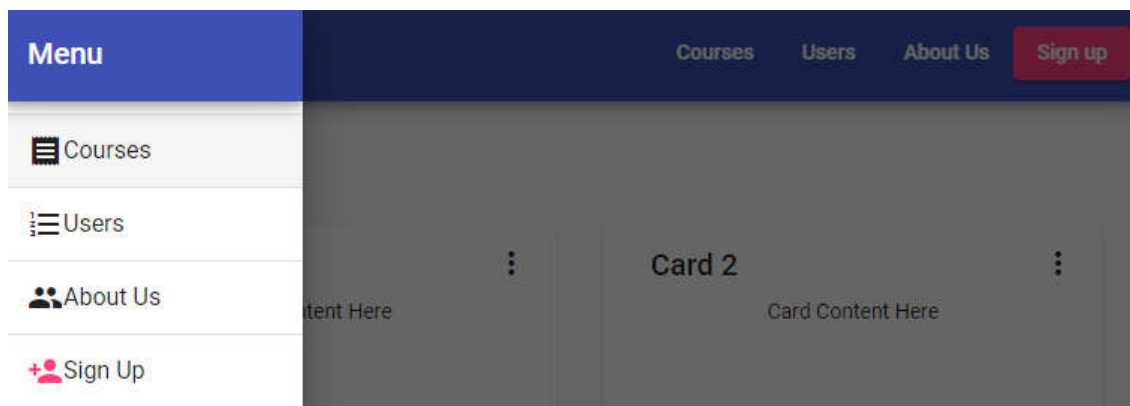
² เลือกรูปไอคอนต่าง ๆ ได้ที่ <https://material.io/resources/icons/?style=baseline>

```

<a mat-list-item href="#">
  <mat-icon aria-hidden="false">receipt</mat-icon>
  Courses</a>
</mat-list-item>
<mat-list-item href="#">
  <mat-icon aria-hidden="false">format_list_numbered</mat-icon>
  Users</a>
</mat-list-item>
<mat-list-item href="#">
  <mat-icon aria-hidden="false">group</mat-icon>
  About Us</a>
</mat-list-item>
<mat-list-item href="#">
  <mat-icon aria-hidden="false" color='accent'>person_add</mat-icon>
  Sign Up</a>
</mat-list-item>
</mat-nav-list>
</mat-sidenav>

```

เมื่อทดสอบเปิดดูรายการนำทางจะได้ผลดังรูปต่อไปนี้



รูป 3 ผลการใส่ไอคอนต่าง ๆ

แต่มีบางอย่างยังไม่เข้าที่เข้าทางคือ เมื่อด้านบนยังคงอยู่ พร้อม ๆ กับเมนูด้านข้าง สิ่งที่เราควรจะเป็นคือ เมื่อนำทางด้านบนควรจะหายไปเมื่อขนาดหน้าจอเล็ก และเมื่อหน้าจอขนาดใหญ่ เมื่อด้านบนควรปรากฏ ดังเหลือแก๊วอีกนิดเดียว คือทำอย่างไรให้เมื่อนำทางด้านบนหายไป

วิธีการใช้การเปลี่ยนแบบจากเมื่อนำทางด้านข้าง ที่เปิด และปิด ตามขนาดหน้าจอได้อย่างไร ด้วยการใช้ คำสั่ง *ngIf ตรวจสอบ (isHandset\$) ของเมนูด้านข้าง ซึ่งจะทำงานตรงกันข้ามกับ เมื่อด้านบน สิ่งที่เราเพิ่มคือ ใช้คำสั่ง *ngIf ซึ่งแทน NOT ดังแก้ไขได้ดังนี้

Code 10. src/app/navigation/navigation.component.html

```

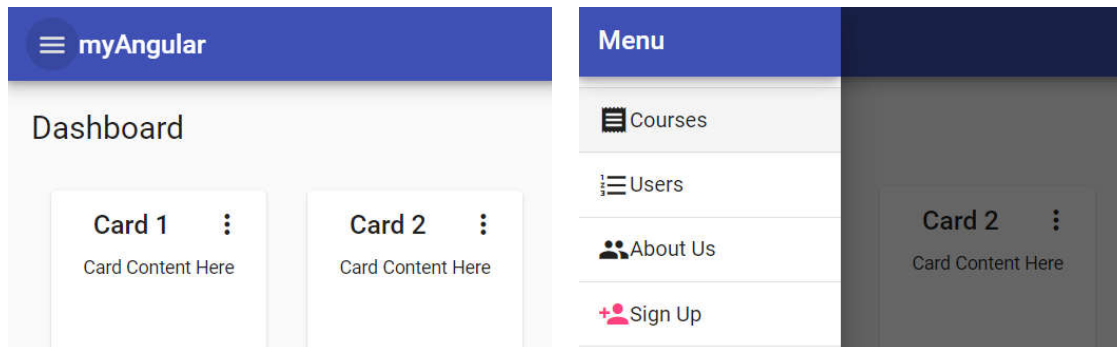
<span>myAngular</span>
<div [style.flex]="1"></div>
<div *ngIf="!(isHandset$ | async)">
  <a mat-button routerLink='courses'>Courses</a>

```

```

<a mat-button routerLink='users'>Users</a>
<a mat-button (click)="openDialog()">About Us</a>
<a mat-raised-button color='accent' routerLink='signup'>Sign up</a>
</div>

```



รูป 4 ผลการเอารายการเมนูด้านบนออก เมื่อจอขนาดเล็ก

สร้างเส้นทาง

ในส่วนเนื้อหาที่ใช้แสดงผล เมื่อคลิกเมนูนำทาง หรือปุ่มคำสั่งต่าง ๆ ทั้งที่อยู่ในส่วนใน <mat-nav-list> และในส่วน <mat-toolbar> ในที่นี้เลือกใช้ **Schematic**: dashboard, table, และ address-form เพื่อแทนคอมโพเนนต์ของ Courses, Users, และ Signup ตามลำดับ สำหรับ About Us สร้างเป็นคอมโพเนนต์ทั่วไป

```

ng g @angular/material:dashboard courses
ng g @angular/material:table users
ng g @angular/material:address-form signup
ng g c aboutus

```

เมื่อได้คอมโพเนนต์ต่าง ๆ ครบแล้ว ต่อมาสร้างเส้นทางใน routing.module.ts โดยเพิ่มทั้งในส่วน import และ อาร์เรย์ของ routes โดยมีเส้นทาง “/courses” เป็นเส้นทางเริ่มต้น

Code 11. src/app/app-routing.module.ts

```

import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { CoursesComponent } from './courses/courses.component';
import { UsersComponent } from './users/users.component';
import { SignupComponent } from './signup/signup.component';
const routes: Routes = [
  {path: 'courses', component: CoursesComponent},
  {path: 'users', component: UsersComponent},
  {path: 'signup', component: SignupComponent},
  {path: '', redirectTo: '/courses', pathMatch: 'full' },
];

```

ใส่ <router-outlet><router-outlet>

ต่อมาสร้างเส้นทาง routLink ทั้งใน <tool-bar> และ <mat-nav-list> แต่เฉพาะส่วน About Us ทำเป็นไดอะล็อก (dialog) ซึ่งในการสร้างใช้โมดูล MatDialogModule ดังจะอธิบายในหัวข้อถัดไป ดังนั้นควรแก้ไขในส่วนการนำทางดังนี้

Code 12. src/app/navigation/navigation.component.html

```
<a mat-button routerLink='courses'>Courses</a>
<a mat-button routerLink='users'>Users</a>
<a mat-button (click)="openDialog()">About Us</a>
<a mat-raised-button color='accent' routerLink='signup'>Sign up</a>
```

ถ้าไม่ใช่ routerLink จะใช้แบบ href ก็ได้เช่นใช้ “/courses” ดังตัวอย่างในรายการนำทางด้านข้าง (<mat-nav-list>) ยกเว้น About Us ทำเป็นไดอะล็อก ซึ่งก็ให้ผลเหมือนกัน

Code 13. src/app/navigation/navigation.component.html

```
<a mat-list-item href="/courses">
  <mat-icon aria-hidden="false">receipt</mat-icon>
  Courses
</a>
<a mat-list-item (click)="openDialog()">
  <mat-icon aria-hidden="false">group</mat-icon>
  About Us</a>
<mat-divider></mat-divider>
```

สร้างไดอะล็อก

ในการใช้งานไดอะล็อกจำเป็นต้องนำเข้าโมดูล MatDialogModule และใส่โมดูลนี้ในอาร์เรย์อาร์เรย์ imports[] ในไฟล์ app.module.ts

Code 14. src/app/app.module.ts

```
import { MatDialogModule } from '@angular/material/dialog';
```

ที่ผ่านมาเราไม่ได้สร้างเส้นทางของคอมโพเนนต์ AboutusComponent โดยตรง แต่เราจะใช้ไดอะล็อกเพื่อเปิดคอมโพเนนต์นี้ โดยจะเปิดผ่านเปิดผ่านคอมโพเนนต์ NavigationComponent จึงต้องนำเข้าส่วนเกี่ยวข้องกับไดอะล็อกผ่านคอมโพเนนต์นี้

Code 15. src/app/navigation/navigation.component.ts

```
import { Component, Inject } from '@angular/core';
import { MatDialog } from '@angular/material/dialog';
import { AboutusComponent } from '../aboutus/aboutus.component';
```

ไดอะล็อกถือเป็นงานบริการอย่างหนึ่งทำให้การใช้งานผ่าน Dependency Injection (DI) จึงใช้งานผ่านการสร้างสมาชิกผ่าน คอนสตรัคเตอร์ (constructor) ได้

Code 16. src/app/navigation/navigation.component.ts

```
constructor(
  private breakpointObserver: BreakpointObserver,
  public dialog: MatDialog){}
```

ที่ผ่านมาเราได้เรียกใช้ฟังก์ชัน openDialog() แต่เรายังไม่เขียนฟังก์ชันนี้เลย เราจะใช้ฟังก์ชันนี้เพื่อเปิดคอมโพเนนต์ AboutusComponent การเปิดจะเปิดผ่านตัวแปร dialog ที่ได้สร้างไว้คอนสตรัคเตอร์

Code 17. src/app/navigation/navigation.component.ts

```

openDialog(): void {
    const dialogRef = this.dialog.open(AboutusComponent,{ width: '600px'});
}

```

เมื่อทดลองคลิกรายการนำทาง About Us เราจะได้ไดอะล็อก ซึ่งเป็นหน้าต่างที่มีความกว้างตามที่ระบุ 600px แต่ความสูงไม่ระบุ ก็จะแสดงตามข้อมูลในคอมโพเนนต์ AboutusComponent ซึ่งยังไม่มีข้อมูลอะไรนอกจากคำว่า “aboutus works!”

โครงสร้างไดอะล็อกทั่วไป ประกอบด้วยส่วน 3 ส่วนคือส่วนหัวใช้คำสั่งไคเร็กทีฟ (directive) mat-dialog-content แทนส่วนเนื้อหาใช้คำสั่งไคเร็กทีฟ mat-dialog-content และส่วนโต้ตอบใช้คำสั่งไคเร็กทีฟ mat-dialog-action

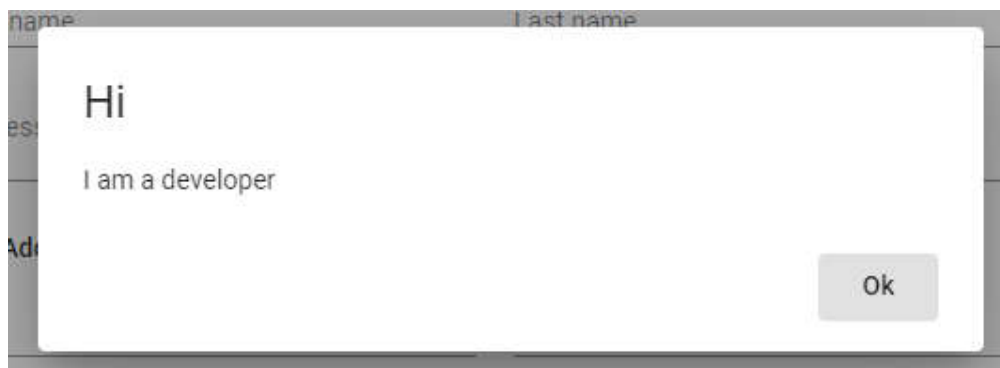
Code 18. src/app/aboutus/aboutus.component.html

```

<h1 mat-dialog-title>Hi</h1>
<div mat-dialog-content>
    <p>I am a developer</p>
</div>
<div mat-dialog-actions align="end">
    <button mat-button mat-dialog-close>Ok</button>
</div>

```

ถึงตอนนี้สามารถทดสอบไดอะล็อก ก็จะได้ข้อมูลเบื้องต้นตามรูปต่อไปนี้



รูป 5 ผลการแสดงไดอะล็อก

จากตัวอย่างที่ผ่านมาไดอะล็อกใช้การเปิดผ่านฟังก์ชัน open() ซึ่งจะคืนค่า MatDialogRef คำนี้นสามารถตัวแปรสามารถดำเนินการหลังการปิดไดอะล็อกได้ ด้วยฟังก์ชัน afterClose() โดยฟังก์ชันนี้รองรับการส่งข้อมูลกลับจากไดอะล็อกได้

ตาราง 1 คำสั่งไคเร็กทีฟของไดอะล็อก

ชื่อคำสั่ง	คำอธิบาย
mat-dialog-title	ประกาศเป็นส่วนชื่อ สามารถใส่อีลีเมนต์ เช่น <h1>
mat-dialog-content	ประกาศเป็นส่วนเนื้อหา
mat-dialog-action	ประกาศเป็นส่วนโต้ตอบได้ เช่น ใส่ ปุ่มกดปิด

mat-dialog-close	ประกาศให้ปิดไดอะล็อก ทำร่วมกับปุ่มคำสั่ง และสามารถส่งผ่านตัวแปรได้ เช่น [mat-dialog-close]="true"
------------------	---------------------------------------------------------------------------------------------------

เรามาลองทดสอบส่งข้อมูลออกจากไดอะล็อก โดยข้อมูลให้ผูกไว้กับ mat-dialog-close และผู้ยกฟังก์ชัน afterClose() ดังแก้ไขหน้าเว็บไดอะล็อกเฉพาะส่วน <button> ดังนี้

Code 19. src/app/aboutus/aboutus.component.html

```
<div mat-dialog-actions align="end">
  <button mat-button [mat-dialog-close]='true' >Ok</button>
</div>
```

และปรับปรุงฟังก์ชัน openFileDialog() โดยเพิ่มฟังก์ชัน afterClose().subscribe() ให้พิมพ์ข้อความที่รับได้ ซึ่งคือค่า 'true' ที่ console.log()

Code 20. src/app/navigation/navigaton.component.ts

```
openDialog(): void {
  const dialogRef = this.dialog.open(AboutusComponent, { width: '600px' });
  dialogRef.afterClosed().subscribe(result => {
    console.log(`Dialog result: ${result}`);
  });
}
```

นอกจากส่งข้อมูลออกจากไดอะล็อกได้แล้ว ยังจะส่งข้อมูลเข้าไดอะล็อกได้ด้วย การส่งข้อมูลเข้ากระทำฟังก์ชัน open() ที่ใช้ในการเปิดไดอะล็อก เช่น ส่งข้อมูล ความกว้าง และข้อมูล JSON ดังแก้ไขฟังก์ชัน open() ใหม่

Code 21. src/app/navigation/navigaton.component.ts

```
const dialogRef = this.dialog.open(AboutusComponent,
  { width: '600px', data:{name:'Monchai'}}
);
```

ฝั่งรับข้อมูลซึ่งก็คือไดอะล็อก ต้องเรียกใช้ตัวรับชื่อ MAT_DIALOG_DATA ซึ่งต้องนำเข้า และใส่ตัวรับนี้ในคอนสตรัคเตอร์ พร้อมกับ รูปแบบข้อมูลที่ได้รับ โดยรูปแบบข้อมูลสร้างเป็นอินเทอร์เฟซ โดยแก้ไขคอมโพเนนต์ AboutusComponent ใหม่ดังนี้

Code 22. src/app/aboutus/aboutus.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { MAT_DIALOG_DATA } from '@angular/material/dialog';
export interface Developer { name:'' }
@Component({
  selector: 'app-aboutus',
  templateUrl: './aboutus.component.html',
  styleUrls: ['./aboutus.component.css']
})
export class AboutusComponent implements OnInit {
```

```

    constructor(@Inject(MAT_DIALOG_DATA) public data: Developer) { }
    ngOnInit(): void { }
}

```

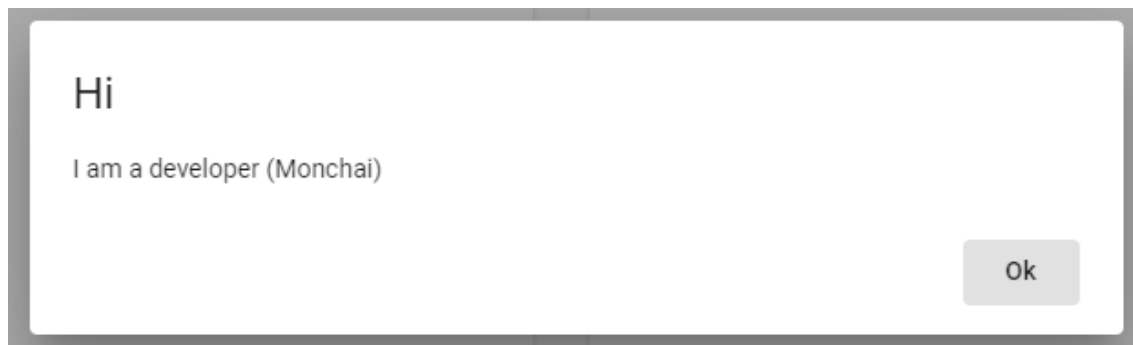
เมื่อรับข้อมูลได้แล้ว สมมติให้แสดงข้อมูลที่รับได้ผ่าน `<div mat-dialog-content>` รูปแบบข้อมูลใช้ `data.name` ในการอ้างอิง ดังการผูกข้อมูลไว้กับ `<p>` จึงแก้ไขเฉพาะส่วน `<p>` ดังนี้

Code 23. `src/app/aboutus/aboutus.component.html`

```

<div mat-dialog-content>
  <p>I am a developer ({{data.name}})</p>
</div>

```



รูป 6 ผลการแสดงโต้ตอบที่รับข้อมูลเข้ามา

สร้างฟอร์ม


สิ่งแรกๆ ที่เราควรเรียนรู้การสร้างฟอร์มให้อยู่ในรูปแบบ การ์ด (Card) ซึ่งบรรจุฟอร์มในรูปแบบโครงสร้างของ `<mat-card-xxx>` ต่างๆ

ตาราง 2 อีลีเมนต์ต่าง ๆ ของ Card

ชื่ออีลีเมนต์	คำอธิบาย
<code><mat-card-title></code>	ส่วนแสดงชื่อการ์ด
<code><mat-card-subtitle></code>	ส่วนแสดงชื่อรอง
<code><mat-card-content></code>	ส่วนแสดงเนื้อหา
<code><mat-card-image></code>	ส่วนแสดงภาพ
<code><mat-card-action></code>	ส่วนแสดงปุ่มคำสั่ง
<code><mat-card-footer></code>	ส่วนแสดงข้อมูลท้ายสุด

จากฟอร์มที่สร้างมาให้ เราต้องการแก้ไขให้มีรูปแบบเพื่อลงทะเบียนผู้ใช้งาน ดังนั้นข้อมูลที่เราต้องมีคือ ชื่อ ชื่อสกุล เพศ สาขาวิชา วันเดือนปีเกิด และอีเมล โดยให้ เพศแสดงเป็นปุ่มเรดิโอ (Radio button) สาขาวิชาแสดงเป็นปุ่มอ็อปชัน (Option) หรือหลายตัวเลือก วันเดือนปีแสดงเป็นรูปปฏิทิน ดังหน้าเว็บนี้ที่ต้องการควมเรียงการดังรูปต่อไปนี้

Sign up

First name	Last name
example@email.com	Major
Gender	Birthday
<input type="radio"/> Male <input type="radio"/> Female	Choose a date 

Submit

รูป 7 หน้าเว็บ Sign up

จากรูปนี้จะมีใช้ปฏิทิน ทำให้ต้องนำเข้าโมดูลสำคัญสองตัว MatDatepickerModule และ MatNativeDateModule เมื่อนำเข้าแล้วต้องเพิ่มโมดูลเหล่านี้ลงอาร์เรย์ imports : [] ด้วย

Code 24. src/app/app.module.ts

```
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatNativeDateModule } from '@angular/material/core';
```

ก่อนที่จะแก้ไขหน้าเว็บ มาสร้างตัวแปร user เพื่อผูกข้อมูลกับหน้าเว็บ โดยใช้ตัวแปรนี้สร้างแทน FormBuilder รองรับข้อมูลแบบกลุ่ม นอกจากนี้ยังสร้างตัวแปร majors แทนข้อมูลแบบหลายตัวเลือก

ในตัวอย่างนี้ใช้การแสดงวันที่ในรูปแบบที่ต้องการเลยต้องนำเข้า formatDate ให้กับคอมโพเน้นท์นี้ด้วย

```
import { formatDate } from '@angular/common';
```

Code 25. src/app/signup/signup.component.ts

```
export class SignupComponent {
  user = this.fb.group({
    firstName: [null, Validators.required],
    lastName: [null, Validators.required],
    gender: [null, Validators.required],
    email: [null, Validators.compose([
      Validators.required, Validators.email])
    ],
  ],
```

```

    major: [null, Validators.required],
    birthday: [null, Validators.required],
  });
  majors = [
    {name: 'Information Technolog', abbreviation: 'IT'},
    {name: 'Business Computer', abbreviation: 'BC'},
    {name: 'Other', abbreviation: 'OT'},
  ];
  constructor(private fb: FormBuilder) {}
  getErrorMessage() {
    if (this.user.controls['email'].hasError('required')) {
      return 'You must enter a value';
    }
    return this.user.controls['email'].hasError('email') ? 'Not a valid email':'';
  }
  onSubmit() {
    let mydate = new Date(this.user.value.birthday);
    alert(formatDate(mydate, 'M/dd/yyyy', 'en-US'));
  }
}

```

สำหรับฟังก์ชัน `getErrorMessage()` สร้างเพื่อรองรับการทำงานผิดพลาดสองแบบคือ ความผิดพลาดจากการไม่กรอกข้อมูลเลย และความผิดพลาดจากไม่อยู่ในรูปแบบอีเมล และฟังก์ชัน `onSubmit()` ใช้ทดสอบการอ่านตัวแปรที่ผู้ใช้กรอก เราจะทดลองให้แสดงค่าต่าง ๆ ผ่านหน้าต่างแจ้งเตือน (`alert`)

ต่อไปก็ปรับแต่งหน้าเว็บ โดยเฉพาะข้อมูลในส่วน `<mat-card-content>` บรรจุ `<mat-form-field>` ซึ่งผูกกับ `formControlName` ทุกตัว สิ่งที่น่าสังเกตคือในส่วนการตรวจสอบความผิดพลาดของอีเมล ใช้คำสั่ง `invalid` เพื่อรองรับความผิดพลาดทุกชนิด

Code 26. `src/app/signup/signup.component.html`

```

<form [formGroup]="user" novalidate (ngSubmit)="onSubmit()">
  <mat-card class="shipping-card">
    <mat-card-header>
      <mat-card-title>Sign up</mat-card-title>
    </mat-card-header>
    <mat-card-content>
      <div class="row">
        <div class="col">
          <mat-form-field class="full-width">
            <input matInput placeholder="First name" formControlName="firstName">
            <mat-error *ngIf="user.controls['firstName'].hasError('required')">
              First name is <strong>required</strong>
            </mat-error>
          </mat-form-field>
        </div>
        <div class="col">
          <mat-form-field class="full-width">

```

```

        <input matInput placeholder="Last name" formControlName="lastName">
        <mat-error *ngIf="user.controls['lastName'].hasError('required')">
            Last name is <strong>required</strong>
        </mat-error>
    </mat-form-field>
</div>
</div>
<div class="row">
    <div class="col">
        <mat-form-field class="full-width">
            <input matInput placeholder="example@email.com" formControlName="email">
            <mat-error *ngIf="user.controls['email'].invalid">
                <string>{{getErrorMessage()}}</string>
            </mat-error>
        </mat-form-field>
    </div>
    <div class="col">
        <mat-form-field class="full-width">
            <mat-select placeholder="Major" formControlName="major">
                <mat-option *ngFor="let major of majors" value= {{major.name}}>
                    {{ major.name }}
                </mat-option>
            </mat-select>
            <mat-error *ngIf="user.controls['major'].hasError('required')">
                Major is <strong>required</strong>
            </mat-error>
        </mat-form-field>
    </div>
</div>
<div class="row">
    <div class="col">
        <label>Gender</label>
        <mat-radio-group aria-label="Select an option" formControlName="gender">
            <div class='row'>
                <mat-radio-button value="male">Male</mat-radio-button>
                <mat-radio-button
                    style="margin-left: 20px" value="female">Female</mat-radio-button>
            </div>
        </mat-radio-group>
    </div>
    <div class='col'>
        <div class='col'>
            <label>Birthday</label>
        </div>
        <mat-form-field appearance="fill">
            <mat-label>Choose a date</mat-label>
            <input matInput [matDatepicker]="picker" formControlName='birthday'>
            <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>

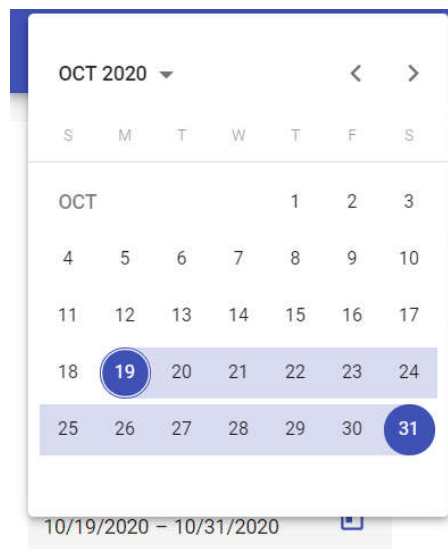
```

```

        <mat-datepicker #picker></mat-datepicker>
      </mat-form-field>
    </div>
  </div>
</mat-card-content>
<mat-card-actions align='end'>
  <button mat-raised-button color="primary" type="submit">Submit</button>
</mat-card-actions>
</mat-card>
</form>

```

หน้าต่างปฏิทินแบบเลือกระยะเวลา เป็นคุณสมบัติหนึ่งที่นำแนะนำให้ใช้ จากเดิมต้องทำสองหน้าต่าง แขนแต่ละช่วงวันเวลา



รูป 8 หน้าต่างปฏิทินแบบเลือกระยะได้

ด้วยการใช้ `<mat-date-range-input>` และ `<mat-date-range-picker>` ในตัวอย่างที่ผ่านมาได้ใช้ปฏิทินแบบเลือกได้ช่วงเดียวและใส่ชื่ออ้างอิง picker ไปแล้ว ถ้าจะทดลองทำหน้าต่างปฏิทินแบบใหม่นี้ต้องเปลี่ยนชื่อไม่ให้ซ้ำ ในตัวอย่างต่อไปนี้จะใช้อ้างอิงว่า `mypicker`

Code 27. `src/app/signup/signup.component.html`

```

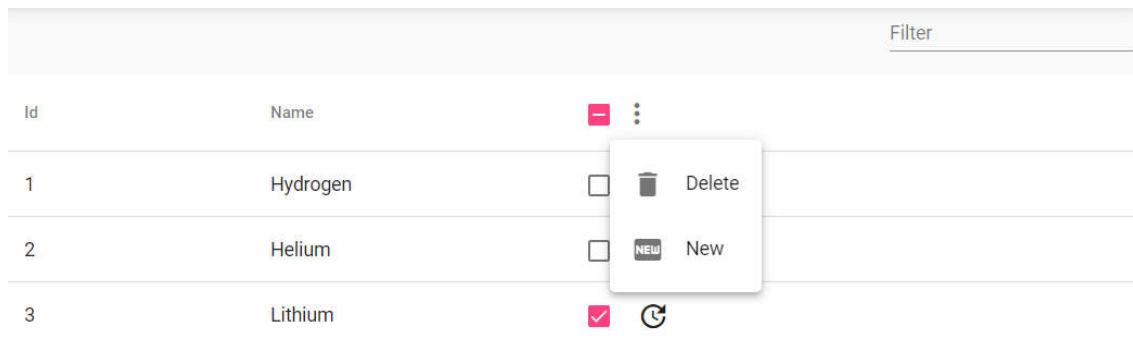
<mat-form-field appearance="fill">
  <mat-label>Enter a date range</mat-label>
  <mat-date-range-input [rangePicker]="mypicker">
    <input matStartDate placeholder="Start date">
    <input matEndDate placeholder="End date">
  </mat-date-range-input>
  <mat-datepicker-toggle matSuffix [for]="mypicker"></mat-datepicker-toggle>
  <mat-date-range-picker #mypicker></mat-date-range-picker>
</mat-form-field>

```

สร้างตาราง

การสร้างตารางจากเมทเทอร์เรียลไคซ์ ถือว่ามีคุณสมบัติโดดเด่นมาก ให้ใช้กับข้อมูลไว้ครบครัน ไม่ว่าจะเป็นการเรียงข้อมูล การลบ การเพิ่ม การสร้างจำนวนการเลือกต่อหน้าว่าจะให้มีกี่แถว รวมถึงการสร้างการสืบค้น ที่ลดเวลาการเขียนโปรแกรมด้วยตนเองได้มาก

ในการทดลองสร้างเราจะเลือกสร้างในคุณสมบัติต่าง ๆ ที่ตารางของเมทเทอร์เรียลไคซ์ มีให้ใช้ โดยหน้าตารางที่จะสร้างมีหน้าตาดังนี้



Id	Name
1	Hydrogen
2	Helium
3	Lithium

รูป 9 หน้าตาตารางที่ต้องการ

ตาราง 3 อีลีเมนต์ต่าง ๆ ของตาราง

ชื่ออีลีเมนต์	คำอธิบาย
<table mat-table>	ประกาศใช้ตาราง ซึ่งเป็นส่วนกรอบนอกสุด
<ng-container matColumnDef="column_name">	ประกาศเพื่อเก็บรายละเอียดคอลัมน์ ที่ประกอบด้วย <th> และ <td> ซึ่งมีอื่นประกอบด้วย เช่น ปุ่มคำสั่ง
<th mat-header-cell>	ประกาศแทนชื่อคอลัมน์
<td mat-cell>	ประกาศข้อมูลในคอลัมน์
<tr mat-header-row>	ประกาศลำดับการแสดงผลของคอลัมน์ ในแถวแรก
<tr mat-row>	ประกาศข้อมูลในแต่ละแถว

ตารางมี 3 ส่วนประกอบหลักคือ การกำหนดแหล่งข้อมูลที่จะนำมาแสดง การกำหนดรูปแบบการแสดงผลของคอลัมน์ และแถว

การเพิ่มตารางเริ่มด้วยการ สร้างอีลีเมนต์ <table mat-table> และกำหนดแหล่งข้อมูล (dataSource) ซึ่งอยู่ในรูปแบบอาร์เรย์ ข้อมูลนี้จะถูกตรวจสอบอัตโนมัติ ไม่ว่าเป็นการแทรกแถวใหม่ การลบ การปรับปรุง

```
<table mat-table [dataSource]="myDataArray">
  ...
</table>
```

ต่อจากนั้น ก็กำหนดรูปแบบของคอลัมน์ให้แต่ละคอลัมน์ โดยกำหนดชื่อคอลัมน์ที่ใช้แทนแถวแรก และแถวแทนข้อมูลในตาราง การหนดชื่อคอลัมน์ใช้ matColumnDef ซึ่งจะต้องชื่อที่ไม่ซ้ำกับคอลัมน์อื่น ภายในอีลีเมนต์ <ng-container>

จะต้องกำหนดรูปแบบการแสดงผลของแถวแรก ด้วยอิลีเมนต์ `<th mat-header-cell>` ภายในอิลีเมนต์นี้จะเขียนข้อมูลที่จะแสดงผล และอิลีเมนต์ `<td mat-cell>` ใช้แทนคอลัมน์ในแถวข้อมูล

```
<!-- Id Column -->
<ng-container matColumnDef="id">
  <th mat-header-cell *matHeaderCellDef mat-sort-header>Id</th>
  <td mat-cell *matCellDef="let row">{{row.id}}</td>
</ng-container>
```

และสุดท้ายแทนข้อมูลแต่ละแถว ด้วย `<tr>` โดย `<tr>` แรกแทนลำดับข้อมูล และ `<tr>` แทนข้อมูลที่แสดง

```
<tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
<tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
```

MatTableDataSource

สก็แมตริกส์ตารางที่สร้างมาให้ไม่ได้ใช้ `MatTableDataSource` ซึ่งเป็น API ที่รวมหลายฟังก์ชันสำหรับดำเนินการกับตาราง เช่น การเรียง การสืบค้นข้อมูล โดยสก็แมตริกส์ตารางนี้เลือกที่สร้าง `users-datasource.ts` มาให้ ซึ่งแน่นอนว่าทำให้ไม่ครบทุกฟังก์ชัน ใน `MatTableDataSource` ดังนั้นเพื่อความสะดวกที่ไม่ต้องเขียนฟังก์ชันมาใช้เอง เราควรเปลี่ยนมาใช้ API แทน โดยนำเข้า API นี้ก่อน

Code 28. src/app/users/users.component.ts

```
import { MatTableDataSource } from '@angular/material/table';
import { UsersDataSource, UsersItem, EXAMPLE_DATA } from './users-datasource';
```

จากตัวอย่างการนำนี้ ได้นำเข้า `EXAMPLE_DATA` มาด้วย เพราะไม่ต้องการใช้งาน `UserDataSource` แล้ว แต่จะใช้ `MatTableDataSource` แทน หลังจากนั้นดัดแปลงข้อมูลบางอย่างในไฟล์ `users-datasource.ts` โดยเพิ่มคีย์เวิร์ด `export` หน้าตัวอย่างข้อมูลด้วย

```
export const EXAMPLE_DATA: UsersItem[]
```

ต่อมาต้องกำหนด `dataSource` เป็นไทป์ `MatTableDataSource<>` ใส่ค่าเจเนอริก (generic) เป็น `UsersItem` ที่ได้จากนำเข้า และแก้ไขให้ `dataSource` มาจากออบเจกต์ `MatTableDataSource`

Code 29. src/app/users/users.component.ts

```
dataSource: MatTableDataSource<UsersItem>;
ngOnInit(): void {
  this.dataSource = new MatTableDataSource(EXAMPLE_DATA);
}
```

เพิ่มการสืบค้นให้ตาราง

สก็แมตริกส์ตารางที่สร้างมาให้ไม่ได้มี ส่วนการสืบค้นมาให้ ดังนั้นจะต้องสร้างส่วนนี้ขึ้นมาเอง โดยเพิ่มในส่วนบนของตาราง และใส่ฟังก์ชัน `applyFilter()` เพื่อรองรับเหตุการณ์การกดคีย์บอร์ดขึ้นทุกครั้ง

Code 30. src/app/users/users.component.html

```
<mat-form-field style='float:right;margin-right:5px'>
  <mat-label>Filter</mat-label>
  <input matInput (keyup)="applyFilter($event)" placeholder="Ex. ium" #input>
</mat-form-field>
```

Code 31. src/app/users/users.component.ts

```
applyFilter(event: Event): void {
  const filterValue = (event.target as HTMLInputElement).value;
  this.dataSource.filter = filterValue.trim().toLowerCase();

  if (this.dataSource.paginator) {
    this.dataSource.paginator.firstPage();
  }
}
```

เพิ่มปุ่ม ลบ-ปรับปรุง-แทรก

เราจะเพิ่มปุ่ม ที่ทำงานกับข้อมูล ทั้ง ลบ ปรับปรุง และแทรกข้อมูลใหม่ โดยมีแนวคิดให้ เช็กบ็อก (checkbox) เพื่อเลือกข้อมูลแถวใด (เลือกได้หลายแถว) เพื่อให้ ลบได้เท่านั้น ส่วนการแทรกข้อมูลใหม่ กับ การปรับปรุงข้อมูล ทำได้ทีละแถว ดังนั้นการเพิ่มนี้จะสร้างขึ้นเป็นคอลัมน์ใหม่ ตั้งชื่อคอลัมน์ว่า select

Code 32. src/app/users/users.component.html

```
<!-- Select Column -->
<ng-container matColumnDef="select">
  <th mat-header-cell *matHeaderCellDef>
    <mat-checkbox (change)="$event ? masterToggle() : null"
      [checked]="selection.hasValue() && isAllSelected()"
      [indeterminate]="selection.hasValue() && !isAllSelected()"
      [aria-label]="checkboxLabel()">
    </mat-checkbox>
    <button mat-icon-button [matMenuTriggerFor]="menu"
      aria-label="Example icon-button with a menu">
      <mat-icon>more_vert</mat-icon>
    </button>
    <mat-menu #menu="matMenu">
      <button mat-menu-item (click)="deleteUsers()">
        <mat-icon>delete</mat-icon>
        <span>Delete</span>
      </button>
      <button mat-menu-item (click)="openDialogAddUser()">
        <mat-icon>fiber_new</mat-icon>
        <span>New</span>
      </button>
    </mat-menu>
  </th>
  <td mat-cell *matCellDef="let row" >
```

```

      <mat-checkbox (click)="$event.stopPropagation()"
        (change)="$event ? selection.toggle(row) : null"
        [checked]="selection.isSelected(row)"
        [aria-label]="checkboxLabel(row)">
    </mat-checkbox>
    <button mat-button matTooltip="update" matTooltipPosition="right"
      (click)="openDialogUpdate(row)">
      <mat-icon>update</mat-icon>
    </button>
  </td>
</ng-container>

```

จากคอลัมน์นี้ได้เพิ่มฟังก์ชัน `deleteUsers()` ซึ่งได้จากการเลือกรายการตาม `<mat-checkbox>` ในกรณีรายการใดถูกเลือกก็จะถือว่ารายการนั้นจะถูกลบเมื่อกดปุ่มลบ ดังนั้นแล้วจึงต้องสร้างฟังก์ชันเพื่อลบข้อมูล

Code 33. `src/app/users/users.component.ts`

```

deleteUsers(): void{
  this.dataSource.data.forEach((row, key) => {
    if (this.selection.isSelected(row)){
      this.dataSource.data.splice(key, 1);
      this.dataSource._updateChangeSubscription(); //update dataSource
    }
  });
}

```

จากฟังก์ชันนี้ใช้การวนซ้ำของ `forEach()` ภายในแลมบ์ดา `row` อ่านค่ารายการใดที่ถูกเลือกก็จะถูกเตือนโดยดูตามค่า `index` ซึ่งเป็นเลขดัชนีหรือลำดับข้อมูลในตาราง และที่สำคัญจะต้องป้จข้อมูลที่ได้เปลี่ยนแปลงด้วยฟังก์ชัน

`_updateChangeSubscription()`

สำหรับรายการปรับปรุง และเพิ่มข้อมูล ตั้งใจจะสร้างเป็นหน้าต่างลอยขึ้นมาหรือไดอะล็อก แต่ภายในไดอะล็อกใส่เป็นฟอร์มแทน ดังนั้นใช้สก็แมตติกส์แบบ `address-form`

```

ng g @angular/material:address-form updateUser
ng g @angular/material:address-form addUser

```

เราเริ่มจากปรับปรุงข้อมูลกัน เมื่อได้สร้างคอมโพเนนต์ `UpdateUserComponent` ซึ่งใช้สร้างจาก สก็แมตติก `address-form` ก็ได้ฟอร์มตามที่ต้องการ จึงปรับปรุงหน้า HTML ใหม่ดังนี้

Code 34. `src/app/update-user/update-user.component.html`

```

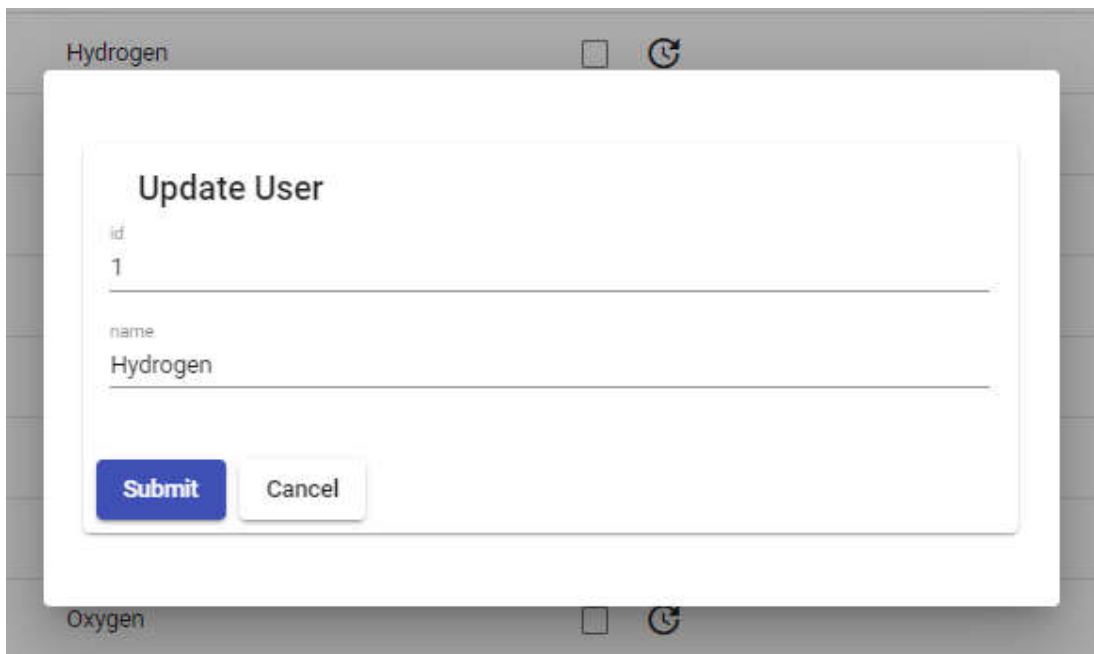
<form [formGroup]="addressForm" novalidate>
  <mat-card class="shipping-card">
    <mat-card-header>
      <mat-card-title>Update User</mat-card-title>
    </mat-card-header>
    <mat-card-content>
      <div class="row">

```

```

<div class="col" >
  <mat-form-field class="full-width">
    <input matInput placeholder="id" [readonly]=true formControlName="id">
  </mat-form-field>
</div>
</div>
<div class="row">
  <div class="col">
    <mat-form-field class="full-width">
      <input matInput placeholder="name" formControlName="name">
    </mat-form-field>
  </div>
</div>
</mat-card-content>
<mat-card-actions>
  <button mat-raised-button color="primary"
    (click)="onSubmit()">Submit</button>
  <button mat-raised-button (click)="dialogClose()">Cancel</button>
</mat-card-actions>
</mat-card>
</form>

```



รูป 10 หน้าต่างการปรับปรุงข้อมูล

เนื่องจาก ข้อมูลมีเพียง id และ name จึงสร้างฟอร์มให้มีอินพุตเพียงสองตัว โดยสมมติให้ตัวอินพุตแรกให้อ่านได้อย่างเดียว โดยกำหนดเป็น [readonly] ส่วนการส่งข้อมูลของฟอร์มจะส่งผ่านฟังก์ชัน onSubmit() ซึ่งต่างจากตัวอย่างการทำไดอะล็อกที่ผ่านมา ที่ส่งจากไฟล์นี้ หรือ HTML โดยตรง สำหรับฟังก์ชัน dialogClose() ทำหน้าที่เพียงปิดหน้าต่าง ไออะล็อกนี้เท่านั้น

สำหรับไฟล์ TS ของคอมโพเน้นท์นี้ มีทั้งรับค่าข้อมูล id และ name จากคอมโพเน้นท์ UsersComponent จึงการนำเข้า Inject เพื่อรับข้อมูลเข้ามา ซึ่งต้องกำหนด อินเทอร์เฟส User เป็นตัวแทนข้อมูลที่รับเข้ามา นอกจากนี้ยังมีการส่งข้อมูลออกไปยังคอมโพเน้นท์ UsersComponent จึงต้องนำเข้า MatDialogRef การส่งข้อมูลด้วยฟังก์ชัน onSubmit() ภายในฟังก์ชันนี้ส่งผ่าน ฟังก์ชัน close(this.addressForm.value) ข้อมูลที่ออกไปจะเป็นรูปแบบ JSON ในขณะที่ฟังก์ชัน dialogClose() ก็มีการเรียกใช้ฟังก์ชัน close() แต่ไม่มีตัวแปรใดส่งไปกับฟังก์ชัน close() นี้

Code 35. src/app/update-user/update-user.component.ts

```
import { Component, Inject } from '@angular/core';
import { FormBuilder, Validators } from '@angular/forms';
import { MAT_DIALOG_DATA, MatDialogRef } from '@angular/material/dialog';
export interface User { id: null; name: null; }
@Component({
  selector: 'app-update-user',
  templateUrl: './update-user.component.html',
  styleUrls: ['./update-user.component.css']
})
export class UpdateUserComponent {
  addressForm = this.fb.group({
    id: [this.data.id],
    // id: [this.data.id, {value: this.data.id, disabled: true}], //1
    name: [this.data.name, Validators.required],
  });

  constructor(
    private fb: FormBuilder,
    private dialogRef: MatDialogRef<UpdateUserComponent>,
    @Inject(MAT_DIALOG_DATA) public data: User) {
    //this.addressForm.get('id').disable(); //2
  }

  onSubmit() {
    console.log(this.addressForm.get('name').value); //3
    this.dialogRef.close(this.addressForm.value);
  }
  dialogClose(){
    console.log("dialog close");
    this.dialogRef.close();
  }
}
```

จากตัวอย่างนี้มีข้อสังเกตหลายประการ ประการแรกการทำให้ อินพุตใดให้อ่านค่าได้อย่างเดียว โดยการ ใส่คำว่า disabled: true ตามที่ท้ายบรรทัดได้ใส่เครื่องหมาย //1 นำหน้าจะผลให้อินพุตนั้นใช้งานไม่ได้เหมือนการทำให้ค่าอ่านได้ อย่างเดียว แต่มีข้อเสียคือ รูปแบบการข้อมูลของ id กับ name มีเหมือนกันจะส่งผลให้ใช้อ่านแบบ addressForm.value

อ่านได้เพียง name เท่านั้น ซึ่งต้องจับรูปแบบใหม่ เช่นต้องใช้การอ่านแบบ บรรทัด //3 ก็จะเสียเวลาจัดรูปแบบจากการอ่านแบบ //3 ส่วนการใช้งานแบบบรรทัด //2 ก็จะส่งผลให้การส่งข้อมูลกลับไปยังคอมโพเนนต์ UsersComponent ไม่ได้

สำหรับคอมโพเนนต์ UsersComponent เมื่อส่งข้อมูลไปแล้ว จะได้ข้อมูลตอบกลับให้ปรับปรุงรายการตามเลข id แต่ก่อนการปรับปรุงต้องตรวจสอบก่อนว่ามีการส่งข้อมูลกลับมาหรือไม่ เพราะหากจำได้ การปิดหน้าต่างใดอะล็อกได้มีการปิดทั้งการปุ่ม Submit และปุ่ม Cancel ถ้ามีข้อมูลตอบกลับมาจริง ก็จะทำให้การค้นหารายชื่อตามเลข id ผ่านฟังก์ชัน find() ซึ่งเขียนอยู่ในรูปแบบแลมบ์ดา เมื่อค้นได้แล้วก็ปรับปรุงเฉพาะชื่อ

Code 36. src/app/users/users.component.ts

```
constructor(public dialog: MatDialog){}
openDialogUpdate(row: UsersItem): void {
  const dialogRef = this.dialog.open(UpdateUserComponent,
    { width: '600px', data:{id: row.id, name: row.name}});

  dialogRef.afterClosed().subscribe(result => {
    console.log(`Dialog result: ${JSON.stringify(result)}`);
    if(result!=null){
      let user = this.dataSource.data.find(u => u.id == result.id);
      user.name = result.name;
    }
  });
}
```

เมื่อทำการปรับปรุงข้อมูลได้ การทำแทรกข้อมูลใหม่ย่อมได้ไม่ยาก โดยการเพิ่มนี้สมมติให้เพิ่มแต่ชื่อ แต่รหัสให้นับเพิ่มจากข้อมูลเดิม เช่น ข้อมูลเดิมมีอยู่ 20 และรหัสประจำชื่อใหม่ควรเป็น 21

เราใช้ฟอร์มของการปรับปรุงข้อมูลแทนได้เลย เพียงแต่ลบส่วนอินพุตของ id ออก และฟังก์ชันก็ใช้ชื่อเดิมคือ onSubmit() และ dialogClose() เนื้อในสองฟังก์ชันก็แทบไม่ได้เปลี่ยนแปลงอะไรมากนัก

Code 37. src/app/add-user/add-user.component.ts

```
import { Component } from '@angular/core';
import { FormBuilder, Validators } from '@angular/forms';
import { MatDialogRef } from '@angular/material/dialog';

@Component({
  selector: 'app-add-user',
  templateUrl: './add-user.component.html',
  styleUrls: ['./add-user.component.css']
})
export class AddUserComponent {
  addressForm = this.fb.group({
    name: [null, Validators.required],
  });

  constructor(
```

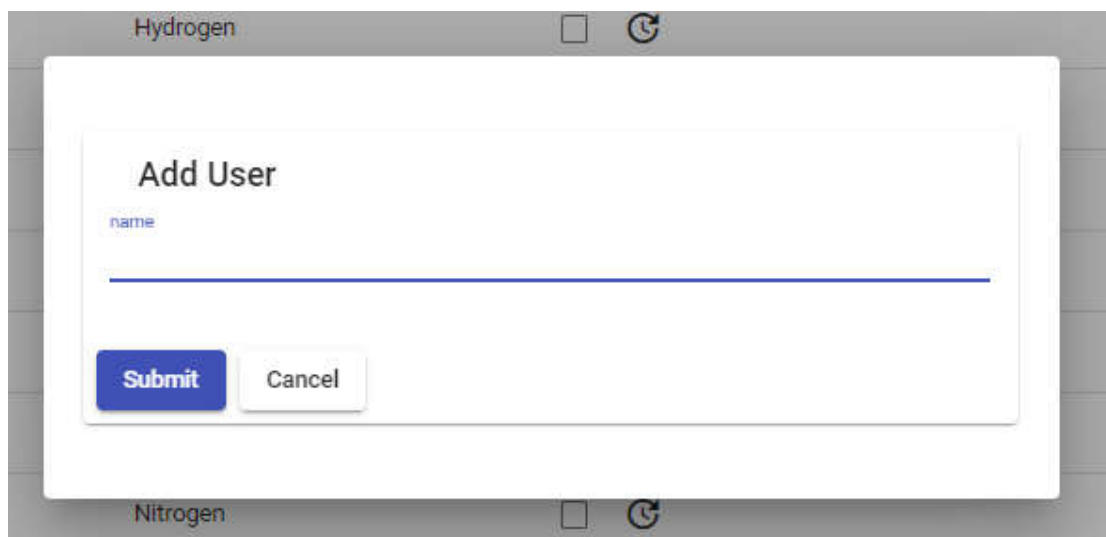
```

private fb: FormBuilder,
private dialogRef: MatDialogRef<AddUserComponent>) {}

onSubmit(): void {
  this.dialogRef.close(this.addressForm.value);
}
dialogClose(): void{
  console.log('dialog close');
  this.dialogRef.close();
}
}

```

จากตัวอย่างนี้ใช้เพียงการนำเข้า MatDialogRef เพื่อใช้เป็นตัวอ้างอิงในการส่งข้อมูลกลับไปยังคอมโพเนนท์ UsersComponent เท่านั้น



รูป 10 หน้าต่างการแทรกข้อมูล

สำหรับ UsersComponent รับข้อมูลมาได้ก็ทำการแทรกด้วยฟังก์ชัน push() แต่ก่อนที่จะแทนให้ทำการอ่านข้อมูลสุดท้ายก่อนเพื่อให้ทราบ id สุดท้าย แล้วทำการบวกเพิ่มไปหนึ่งค่า เมื่อแทรกข้อมูลแล้วต้องปรับปรุง dataSource ด้วย ฟังก์ชัน updateChangesSubscription()

Code 38. src/app/users/users.component.ts

```

openDialogAddUser(): void {
  const dialogRef = this.dialog.open(AddUserComponent, { width: '600px' });
  dialogRef.afterClosed().subscribe(result => {
    if(result!=null){
      let nextId: number = this.dataSource.data.length;
      nextId++;
      this.dataSource.data.push({id: nextId, name: result.name});
      this.dataSource._updateChangeSubscription(); // update datasource
    }
  });
}

```

วิทยาลัยเซาธ์อีสท์บางกอก เอกสารประกอบการสอน การเขียนโปรแกรม Angular/AngularJs

```
}  
});  
}
```

ถึงตอนนี้เราได้ทดลองใช้ แองกูลาร์ แมทเทอร์เรียลดีไซน์ มาพอสมควรแล้ว หน้าที่เหลือที่ยังไม่ปรับปรุงอะไรเพิ่มคือ CoursesComponent ที่สร้างมาจากสกีเมติก dashboard ซึ่งจะเหลือไว้ให้ทดลองเพิ่มเติม แมทเทอร์เรียลดีไซน์ อื่นๆ ด้วยตนเอง

สรุป

การใช้งาน แองกูลาร์ แมทเทอร์เรียลดีไซน์ กับงานเว็บไซต์ช่วยลดปัญหา การออกแบบไปได้มา โดยเฉพาะสร้างด้วยสกีเมติก ในรูปแบบต่าง ดังที่เราได้ทดลองใช้ ในสกีเมติก navigation, dashboard, table, address-form โดยเฉพาะการใช้งาน navigation ช่วยลดการออกแบบรายการทำทางได้มาก เพียงแต่เราต้องมาปรับปรุงในแบบที่เราชอบ และ table ก็มีประโยชน์มาก มีส่วนในการทำงานรายการข้อมูลได้ดี รวมทั้งยังทำเลื่อนเลือกจำนวนแถวที่แสดงผลอันนี้ก็ลดเวลาการเขียนโปรแกรมไปได้มากเช่นกัน สำหรับการทำไดอะล็อก ก็แปลกกว่าคอมโพเนนต์อื่น ๆ เพราะการเรียกใช้งานต้องผ่านการเรียกใช้งานผ่านคอมโพเนนต์อื่น อีกทั้งยังต้องทำการรับ-ส่งข้อมูลไปมาระหว่างคอมโพเนนต์ อย่างไรก็ตามยังมี แมทเทอร์เรียลดีไซน์อีกหลายตัวให้เลือกใช้งาน แต่ด้วยตัวอย่างการใช้งานการใช้งานในบทนี้เป็นพื้นฐานในการใช้งาน แมทเทอร์เรียลดีไซน์ อื่นๆต่อไป

คำถามทบทวน

1. สกีเมติกส์คืออะไร
2. คุณสมบัติ mode ของ mat-sidenav ใช้กำหนดอะไร
3. mat-toolbar คือส่วนประกอบใด สกีเมติกส์ navigation
4. ยกตัวอย่างการใช้งาน mat-icon แล้วที่ยกตัวอย่างเป็นรูปอะไร
5. การที่คอมโพเนนต์หนึ่งต้องการใช้งานไดอะล็อก เริ่มต้นต้องทำอะไร
6. การรับข้อมูลจากไดอะล็อกต้องทำอะไร
7. mat-table ผู้ถูกข้อมูลผ่านตัวแปรอะไร
8. ข้อมูลที่ผูกกับ mat-table เป็นโครงสร้างเป็นแบบใด
9. หลังจากปรับข้อมูลใน mat-table จะต้องเรียกใช้ฟังก์ชันใดทุกครั้ง
10. ให้เขียนรายการ ของแมทเทอร์เรียลดีไซน์ ที่ใช้ในบทนี้ มา สิบรายการ

แบบฝึกหัด

1. สำหรับหน้าต่างเพิ่มผู้ใช้งาน ยังไม่ได้แสดงตัวอย่างโปรแกรม ให้ทดลองสร้างขึ้นในแบบของตนเอง
2. ให้เลือกใช้งาน แมทเทอร์เรียลดีไซน์ อื่น ๆ ที่ไม่ได้ยกตัวอย่างการทำงานในบทนี้ มาใช้งาน โดยเพิ่มเป็นหน้าเว็บเป็นหน้าใหม่ ใส่ แมทเทอร์เรียลดีไซน์ ที่คิดว่าเหมาะเพื่อแสดงผล
3. หน้าที่เหลือที่ยังไม่ปรับปรุงอะไรเพิ่มคือ CoursesComponent ที่สร้างมาจากสกีเมติก dashboard ซึ่งจะเหลือไว้ให้ทดลองเพิ่มเติม แมทเทอร์เรียลดีไซน์ ที่แสดงผลเป็น ชื่อวิชา พร้อมมีรูปภาพประกอบวิชา