การเคลื่อนไหว

การเคลื่อไหวเป็นประสบการณ์สร้างความน่าสนใจ ในการสร้างเว็บไซต์ ผู้งานใช้งานเว็บไซต์จะรู้สึกถึงความเป็นมืออาชีพ นอกจากเป้าหมายจริงในการใช้งานเว็บไซต์ ประสบการณ์ที่ดีต่อเว็บไซต์จึงเป็นอีกส่วนประกอบหนึ่ง ที่สร้างความพึงพอใจต่อผู้ใช้งาน ในบทนี้เราจะมาทำเข้าใจและใช้งานการสร้างการเคลื่อนไหวในแองกูลาร์ ดังจะศึกษาในหัวข้อต่อไปนี้

- สไตย์และการเคลื่อนไหว
- การผูกสถานะเพื่อการเคลื่อนไหว
- การสร้างคีย์เฟรมเพื่อการเคลื่อนไหว
- การสร้างการเคลื่อนไหวระหว่างเป็นเปลี่ยนหน้าเว็บ

การเคลื่อนไหวของแองกูล่าร์ใช้การทำงานร่วมกับ CSS โดยมีเวลาควบคุมการเคลื่อนไหวของแต่ละการเคลื่อนไหวที่เปลี่ยนแปลงตามสไตล์ การเคลื่อนไหวของเองกูล่าร์ได้ถูกรวมอยู่ในระบบหลักแล้วทำให้ไม่ต้องติดตั้งอะไรเพิ่มเติม เพียงแต่นำเข้าโมดูล BrowserAnimationModule ก็สามารถสร้างการเคลื่อนไหวได้

เริ่มต้นใช้งานการเคลื่อนไหว

1. สร้างแอปพลิเคชัน แบบมีเส้นทาง
2. สร้างคอมโพเน้นท์ Home ซึ่งเราจะใช้คอมโพเน้นท์นี้สำหรับการทดลองสร้างการเคลื่อนไหว และสร้างคอมโพเน้นท์ PageNotFound เพื่อใช้เป็นเส้นทางที่หาไม่พบ
3. สร้างเส้นทางเริ่มต้นไปยัง Home ในไฟล์ app-routing.module.ts

```
import { HomeComponent } from './home/home.component';
import { PageNotFoundComponent} from './page-not-found/page-not-
found.component';
const routes: Routes = [
  { path:'home', component: HomeComponent},
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path:'**', component: PageNotFoundComponent },
];
```

4. ใส่เส้นทางในหน้าหลัก (app/app.component.html) ในหน้านี้ให้ลบข้อมูลออกให้หมด ใส่เพียง ข้อมูลต่อไปนี้

```
<router-outlet></ router-outlet>
```

5. เพิ่มนำเข้า BrowserAnimationsModule ในโมดูลหลัก (src/app/app.module.ts)

```
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule
],
```

6. เพิ่มนำเข้าฟังก์ชันด้านการเคลื่อนไหวไปยังคอมโพเน้นท์หลัก (src/app/home/home.component.ts)

```
import {
trigger, state, style, animate, transition, } from '@angular/animations';
```

และเพิ่มคุณสมบัติการเคลื่อนไหวในส่วน @Component

```
@Component({
selector: 'app-root',
templateUrl: 'app.component.html',
styleUrls: ['app.component.css'],
animations: [
//ใช้เขียนต่อในข้อ 7
]
})
```

7. สร้างการเคลื่อนไหว กำหนดจุดเริ่มต้นในชื่อ fly มีสถานะชื่อ intro ในสถานะนี้ได้กำหนดสไตล์ใน 3 ลักษณะคือ padding, bacgroundColor และ transform นอกจากนี้ยังกำหนด การเคลื่อนที่ จากสถานะใด ๆ ไปยังสถานะ out ด้วยเวลาในการเคลื่อนไหว 2 วินาที ดังแทนการเคลื่อนไหวต่อไปนี้ใน animation ของข้อ 6

```
animations: [
       trigger('fly', [
              state('intro', style({
                     padding:'10px',
                     backgroundColor:'#DC143C',
                     transform: 'translateX(5%)' ,
              })),
              transition('* => intro', [
                     animate('2s')
              ]),
       ])
]
```

8. กำหนดการเคลื่อนไหว ที่ src/app/home/home.component.html ให้ชื่อการเคลื่อนไหว [@fly] ตามสถานะ ‘intro’ และ <h2> มีระยะห่างจากทางขวามือ 10%

```
<div style="text-align:center">
  <h2 [@fly]="'intro'" style="margin-right:10%; border-radius:5px">
    Welcome to {{ title }}!
  </h2>
</div>
```

9. บันทึกไฟล์ และดูผลลัพธ์

จากการทดลองใช้งานการเคลื่อนไหวเริ่มต้นที่ได้ทำมา มีการเคลื่อนไหว ใน 3 ลักษณะตามสไตล์ โดยเปรียบเทียบกับการแสดงผลดังเดิมของหน้า HTML

- สไตล์แรก padding เดิมไม่มี ผลการเคลื่อนจะเคลื่อนให้มี padding 10px
- สไตล์ที่สอง backgroundColor เป็นสีพื้นหลัง จากเดิมไม่มี ไปสู่สีแดงเข้ม 100%
- และสไตล์สุดท้าย transform จะทำให้จากเดิมมี margin-right 10% กลายเป็นการเลื่อนจากซ้ายไปขวา (แนวแกน X) อีก 5% ผลทำให้เหลือ 5% (10-5=5%) ทำให้ด้านซ้ายและขวาห่างกันด้านละ 5% เท่ากัน

ตาราง 1 CSS Transform

คุณสมบัติ	ความหมาย	ตัวอย่างการใช้
translate(x,y)	การย้ายตำแหน่งจาก แนวแกน X ไปแนวแกน Y มีค่าตามตัวแปร x, y	transform: translate(10px) transform: translate(10px, 10px)
translateX(x)	การย้ายตำแหน่งจาก แนวแกน X	transform: translate(10px)
translateY(y)	การย้ายตำแหน่งจาก แนวแกน Y	transform: translate(10px)
rotate(angle)	การหมุนมีค่าตามมุม	transform: rotate(20deg)
rotateX(angle)	การหมุนตามแนวแกน X	transform: rotateX(20deg)
rotateY(angle)	การหมุนตามแนวแกน Y	transform: rotateY(20deg)

โครงสร้างการทำการเคลื่อนไหว

จากตัวอย่างที่ได้ทดลองทำ โครงสร้างการทำการเคลื่อนไหวจะอยู่ภายใต้ คุณสมบัติ animations: [ ] ดังเขียนเป็นโครงสร้างอย่างย่อได้คือ

```
animations:[
      trigger('trigger name', [
         state('state name', style({})),
         transition('state name => state name', [animation('time')]
      ])
]
```

ในการสร้างการเคลื่อนไหวจำเป็นต้องสร้างฟังก์ชัน trigger( ) ที่กำหนดสถานะ (state) และการเคลื่อนที่ (transition) ในฟังก์ชัน trigger( ) นี้จะต้องมีชื่อของ trigger เพื่อนำไปอ้างอิงในไฟล์ HTML

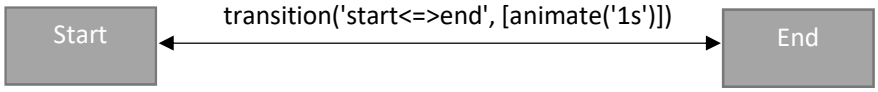
หนึ่งฟังก์ชัน trigger( ) สามารถมีได้หลายสถานะ และได้หลายการเคลื่อนที่ แต่ละสถานะจะกำหนดคุณลักษณะด้วยสไตล์ในรูปแบบ CSS ซึ่งใส่สไตล์ได้มากมาย และแต่ละการเคลื่อนที่จะกำหนดลักษณะการเคลื่อนจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง ด้วยเงื่อนไขการเคลื่อนไหว ในรูปแบบเวลา ซึ่งอาจจะเพิ่มสไตล์ที่กำกับเฉพาะในการเคลื่อนไหวก็ได้

การเคลื่อนไหวตามสถานะและสไตล์

สถานะชื่อหนึ่งจะมีการกำหนดสไตล์ต่าง ๆ การเคลื่อนไหวจากสถานะหนึ่งไปยังอีกสถานะหนึ่งจะทำให้เกิดการเปลี่ยนแปลงสไตล์ การเคลื่อนของสถานะสามารถทำให้เกิดการเคลื่อนแบบทางเดียว หรือสองทาง ด้วยการใช้เครื่องหมาย =>, <=, <=> ได้ เช่นต้องการเคลื่อนที่อยู่ในรูปแบบสองทิศทาง ระหว่างสถานะ start และ end

```
transition('start<=>end', [animate('1s')])
```

ตัวอย่างจาก **รูป 1** กำหนดชื่อสถานะ start มีสไตล์ห่างจากซ้าย 5% และอีกสถานะหนึ่งมีสไตล์ห่างจากซ้าย 85% การเคลื่อนที่ไปมาระหว่างสองสถานะนี้ จะทำให้เกิดการเปลี่ยนแปลงตำแหน่งไปมาซ้าย 5% และ 85% ให้สังเกตว่า มีเพียงสไตล์ left เท่านั้นที่เปลี่ยนแปลง ดังนั้นการเปลี่ยนแปลงจึงเกิดสไตล์ left เท่านั้น ส่วนการกำหนดสไตล์ในที่อื่น เช่น HTML จะไม่มีผลต่อการเปลี่ยนแปลง




```
state('start',
      style({position:'relative',
            left:'5%',
      })
)
```

```
state('end',
      style({position:'relative',
            left:'85%',
      })
)
```

รูป 1 เคลื่อนไหวระหว่างสถานะ start และ end

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

```
state('start', style({position:'relative', left:'5%'})),
state('end', style({position:'relative', left:'85%', })),
transition('start<=>end', [ animate('1s ease-in' )]),
```

เวลาในการเคลื่อนที่

ฟังก์ชัน animate( ) รองรับเวลาในการเคลื่อนที่ 3 ลักษณะคือ เวลาที่ใช้เคลื่อนที่ (duration) เวลาที่หน่วงให้ล่าช้า (delay) และเวลาเร่งหรือชลอ (easing) การกำหนดเวลาสามแบบอยู่ในลักษณะ:

```
animate(duration delay easing)
```

สำหรับตัวแปรแรกจำเป็นต้องมีเสมอ การกำหนดเวลาอยู่ในหน่วยมิลลิวินาที (ms) โดยไม่จำเป็นต้องใส่หน่วยเวลา แต่ก็สามารถใส่หน่วยเวลาได้ ถ้าไม่ต้องการเป็นค่าเวลามิลลิวินาที เช่น ตัวอย่างต่อไปนี้

- กำหนดเป็นมิลลิวินาทีในรูปตัวเลข : 1000
- กำหนดเป็นมิลลิวินาทีในรูปอักษร : '1000ms'
- กำหนดเป็นวินาที : '1s'
- กำหนดเป็นครึ่งวินาที: '0.5s'

ตัวแปรที่สอง เวลาที่หน่วงให้ล่าช้าจากเวลาเริ่มต้น เช่นต้องการกำหนดให้รอ หนึ่งวินาทีก่อนที่จะทำงานในวินาทีถัดไป การกำหนดคือ

```
animate('1s, 1s')
```

และตัวแปรสุดท้ายแทนเวลาเร่งให้เร็วหรือชลอให้ช้า ใช้ควบคุมระหว่างที่มีการเคลื่อนไหว ซึ่งต้องอาศัยคำ ease-xx เป็นตัวกำหนด ตัวอย่างเช่น

- ต้องการรอ หนึ่งวินาที ก่อนทำงาน 2 วินาที และใช้การเร่งในตอนเริ่มต้นแต่ชลอในตอนปลาย: `animate('2s, 1s ease-out')`
- ต้องการรอ หนึ่งวินาที ก่อนทำงาน 2 วินาที และใช้การชลอในตอนเริ่มต้นแต่เร่งในตอนปลาย: `animate('2s, 1s ease-in')`
- ต้องการรอ หนึ่งวินาที ก่อนทำงาน 2 วินาที และใช้การเร่งในตอนเริ่มต้นแต่เร่งในตอนกลางและชลอในตอนปลาย:
  `animate('2s, 1s ease-in-out')`

เพื่อความเข้าใจมากขึ้นของตัวแปรที่สามนี้ ให้พิจารณารูปฟังก์ชัน sine ในสามลักษณะคือ in, out, inOut แทนความเร็วตามการแนวเส้นกราฟ ในแนวแกน x แทนเวลา ในแนวแกน Y ความเร็ว เช่น รูปแรก (easeInSine) ความเร็วในตอนเริ่มต้นน้อย (กราฟมีความชันน้อย) ในตอนปลายมีความเร็วมาก(มีความชันมาก)

รูป 2 เคลื่อนไหวตามกราฟ 3 ลักษณะ (ภาพได้จาก **https://easings.net/**)

ไม่ใช่มีเพียงรูปแบบการเคลื่อนไหวสามแบบนี้เท่านั้นยังมีรูปแบบการเคลื่อนไหวอีกมากมาย สืบค้นรูปแบบการเคลื่อนไหวได้ที่เว็บไซต์ **https://easings.net/**

การผูกสถานะกับ HTML

จากสถานะ start และ end ที่ได้สร้างก่อนหน้านี้ นำมาสร้างการเคลื่อนไหว การแย้งตำแหน่งของกล่องซึ่งสร้างใน HTML ด้วย กำหนดกล่องมีสไตล์เป็นกล่องสีเทา กว้างและสูง 80px ผูกกล่องนี้ด้วยฟังก์ชัน toggleStart( ) และมีผลสถานะตามเงื่อนไข isStart? ถ้าเป็นจริงจะคือค่าสถานะ start แต่ถ้าไม่ใช่จะคืนค่า end

Code 2. src/app/home/home.component.html

```
<div   [@fly]="isStart ?'start' : 'end'"
       (click)='toggleStart()'
       style="background-color:gray;width:80px;height:80px;padding:10px;">
       isStart:<br>{{isStart}}
</div>
```

ฟังก์ชัน toggleStart( ) สร้างในคอมโพเน้นท์ ts ไฟล์ เป็นเพียงจำค่าไปมาระหว่างจริงหรือเท็จ เพื่อรองรับการคลิกจากกล่องของหน้า HTML

Code 3. src/app/home/home.component.ts

```
isStart =true;
toggleStart(){
    this.isStart = !this.isStart;
}
```

สถานะใด ๆ

นอกจากให้ย้ายจากสถานะหนึ่งไปสู่สถานะหนึ่งได้แล้ว ยังสามารถสร้างการย้ายไปสถานะใด ๆ ด้วยสัญลักษณ์ดอกจัน (*) ดังตัวอย่างการใช้:

- start => * เป็นการย้ายจากสถานะ start ไปสู่สถานะใด ๆ

- * => end เป็นการย้ายจากสถานะใด ๆ มาสู่สถานะ end
- * => * เป็นการย้ายไปมาระหว่างสถานะใด ๆ ถ้าในระบบมีสถานเพียง start กับ end จะมีความหมายเหมือน start <=> end

ยังมีการแทนสถานะใดๆ ได้ด้วย void, enter, และ leave การทำงานสามตัวนี้คล้ายกัน กล่าวคือ void แทนเข้าสู่สถานใดหรือออกจากสถานะใด ๆ ดังนั้นแล้วจึงมีสองคำหลังแทนการเข้าและออกจากสถานะใด ๆ ดังนั้นแล้วถ้ามีกำกับให้มีได้เพียงสองสถานะ ตัวอย่างต่อไปนี้มีความหมายเหมือนกัน

```
transition('start<=>end', [ animate('1s ease-in' )]),
```

มีความหมายเหมือนกับ

```
transition('*<=>*', [ animate('1s ease-in' )]),
```

มีความหมายเหมือนกับ

```
transition('void=>*', [ animate('1s ease-in' )]),
transition('*=>void', [ animate('1s ease-in' )]),
```

และมีความหมายเหมือนกับ

```
transition(':enter', [ animate('1s ease-in' )]),
transition(':leave', [ animate('1s ease-in' )]),
```

ต่อไปนี้จะเป็นตัวอย่างการสร้างสถานะใด ๆ โดยใช้การเข้า-ออก สถานะใด ๆ โดยการกำกับสไตล์และการเคลื่อนไหวไว้ด้วยกันของ transition

Code 4. src/app/home/home.component.ts

```
trigger('showHide', [
      transition(':enter', [
                  style({ opacity: 0 }),
                  animate('1s', style({ opacity: 1 })),
      ]),
      transition(':leave', [
                  animate('1s', style({ opacity: 0 }))
      ])
]),
```

ในตัวอย่างนี้ให้ชื่ออ้างอิงเป็น showHide ให้มีสถานะ :enter หรือเข้าสู่สถานะใด ๆ ของ showHide ให้มีความเข้ม (opacity: 0) เป็นศูนย์หรือจางจนมองไม่เห็น แล้วให้แปลงเป็นเข้มจนเห็นเต็มที่ (opacity: 1) และให้เมื่อออกจากสถานะใด ๆ (leave) ให้จางจนมองไม่เห็น โดยทั้งหมดนี้มีเป้าหมายให้แสดง และลบการแสดงนั้นเอง

ต่อไปนิยามอาร์เรย์ users เป็นข้อมูลชื่อสามรายในรูปอาร์เรย์ และให้ฟังก์ชัน disappear( ) ทำหน้าที่ลบข้อมูลในอาร์เรย์ตามเลขลำดับในอาร์เรย์

Code 5. src/app/home/home.component.ts

```
users:String[] =
```

```
        ['Theerapol','Monchai','Chartree','Vinai', 'Vilasinee','Vachira'];
disappear(i:number):void{
        this.users.splice(i,1);
}
```

และสุดท้ายกำหนดการผูกข้อมูลในหน้า HTML ให้แสดงผลข้อมูลทั้งสามรายการ users ในรูปตาราง ใช้ตัวข้อมูลชื่อแต่ละรายด้วยตัวแปร v และข้อมูลลำดับในรูปตัวแปร i นอกจากนี้ยังผูกกับฟังก์ชัน disappear(i) ผ่านตัวแปร i เพื่อนำไปลบรายการข้อมูลในลำดับ i เมื่อคลิกในแถวที่เลือก

Code 6. src/app/home/home.component.html

```
<table class="table">
<tr @showHide
        *ngFor ="let v of users; index as i"
                (click)='disappear(i)'><td>{{v}}</td></tr>
</table>
```

เมื่อเริ่มต้นแสดงผลถือว่าอยู่ในสถานะ enter ทำให้ตารางนี้ค่อยแสดงผลจากภาพจางจนเข้มเต็มที่ในเวลา 1 วินาที และเมื่อคลิกแถว ทำให้ข้อมูลในแถวนั้นถูกลบจึงถือว่าอยู่ในสถานะ leave แถวนั้นจึงค่อย ๆ หายไปจากตาราง

ตาราง 2 สรุปสัญลักษณ์การย้ายสถานะ

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้
=>	การย้ายจากสถานะซ้ายไปสถานะขวา	start => end
<=	การย้ายจากสถานะขวาไปสถานะซ้าย	start <= end
<=>	การย้ายจากสถานะไปมาระหว่างสถานะซ้ายและขวา	start <=> end
*	การย้ายจากสถานะใด เช่น จากสถานะใดๆ ไปสถานะ end	* => end
void	การย้ายเข้าสู่สถานะเมื่อเขียนที่ฝั่งซ้าย หรือย้ายออกจากสถานะ เมื่อเขียนที่ฝั่งขวา เช่น การย้ายเข้าไปสู่สถานะใดๆ	void => *
enter	การย้ายเข้าสู่สถานะ มีความหมายเหมือน void => *	:enter
leave	การย้ายออกจากสถานะ มีความหมายเหมือน * => void	:leave

อ่านเหตุการณ์ของการเคลื่อนไหว

ฟังก์ชัน trigger( ) สามารถปล่อยเหตุการณ์ ($event) ตั้งแต่เริ่มต้นผ่านคุณสมบัติ start จนสิ้นสุดการเคลื่อนไหวได้ผ่านคุณสมบัติ done

ดังนั้นแล้วจะต้องนำเข้า AnimationEvent เพิ่ม

```
import { AnimationEvent } from '@angular/animations';
```

ตัวอย่างต่อไปนี้ได้สร้างฟังก์ชัน onAnimationEvent( ) โดยรับตัวแปร AnimationEvent เพื่ออ่านเหตุการณ์ และอ่านคุณสมบัติในเกตุการณ์ต่าง ๆ การผูกข้อมูลกับตัวแปร eventInfo เพื่อแสดงผลที่หน้าเว็บ รวมทั้งแสดงผลผ่าน console.warn( )

Code 7. src/app/home/home.component.ts

```
eventInfo:String;
```

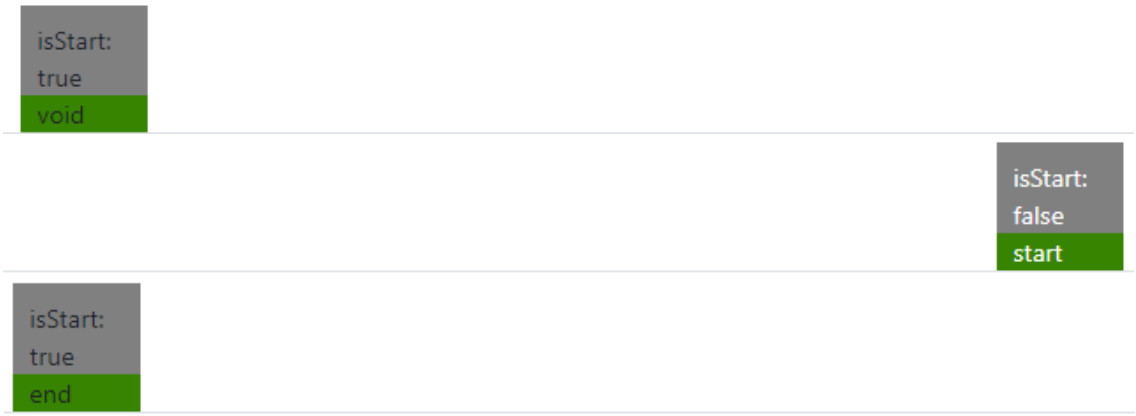
```
onAnimationEvent( event: AnimationEvent ) {
        this.eventInfo = String(event.fromState);

        console.warn(`Animation Trigger: ${event.triggerName}`);
        console.warn(`Phase: ${event.phaseName}`);
        console.warn(`Total time: ${event.totalTime}`);
        console.warn(`From: ${event.fromState}`);
        console.warn(`Element: ${event.element}`);
}
```

ในที่นี้เลือก @fly ที่เคยทำก่อนหน้านี้อ่านเหตุการณ์ ทั้ง start และ done โดยเพิ่ม <p> เพื่อผูกข้อมูลกับตัวแปร eventInfo และดูผลได้จากรูปต่อไป

Code 8. src/app/home/home.component.html

```
<div [@fly]="isStart ?'start' : 'end'" (click)='toggleStart()'
     (@fly.start)="onAnimationEvent($event)"
     ($fly.done)="onAnimationEvent($event)"
      style="background-color:gray;
             width:80px;height:80px;
             padding:10px;color:0;">
      isStart:<br>{{isStart}}
      <p style="background-color:green;
               padding:0 10px;margin:0 -10px">{{eventInfo}}</p>
</div>
```



รูป 3 สถานะเริ่มต้น (บน) เมื่อคลิก 1 ครั้ง (กลาง) และเมื่อคลิกอีกครั้ง (ล่าง)

คีย์เฟรม (Keyframes)

ที่ผ่านมาเราได้ทำให้กล่องเคลื่อนที่ได้จากซ้ายไปขวา การเคลื่อนที่จากจุดเริ่มต้นไปสู่ปลายทางด้วยความเร็วคงที่ เพราะถือว่ามีเฟรมเคลื่อนที่มีระยะที่เท่ากัน แต่ถ้าเฟรมเวลาเคลื่อนที่ได้กำหนดขึ้นใหม่ได้

แองกูล่าร์สามารถสร้างคีย์เฟรมด้วยฟังก์ชัน keyframes( ) ซึ่งกำหนดจำนวนคีย์เฟรมให้มีการเปลี่ยนแปลงแต่ละคีย์เฟรมที่ต่างกันได้ เช่น กำหนด 3 คีย์เยฟรม ก็ให้ใส่สไตล์สามแบบ แต่ก่อนใช้คีย์เฟรมต้องนำเข้า keyframe ด้วย

```
import { keyframes } from '@angular/animations';
```

ตัวอย่างต่อไปนี้สร้างคีย์เฟรม 3 ตัวของกล่องให้เคลื่อนที่จากซ้ายไปขวา แต่กำหนดระยะการเคลื่อนที่ต่างกัน จะส่งผลต่อความเร็วในการเคลื่อนแต่ละระยะไม่เท่ากัน เนื่องจากการเคลื่อนไปซ้าย และไปขวาใช้ระยะที่ต่าง ๆ กัน จึงต้องสร้างการเคลื่อนไปซ้ายและขวาทั้งสองแบบ

Code 9. src/app/home/home.component.ts

```
transition('start=>end', [ animate('2s ease-in',
      keyframes([
                  style({position:'relative', left:'5%'}),
                  style({position:'relative', left:'70%'}),
                  style({position:'relative', left:'85%'}),
      ])
)]),
transition('end=>start', [ animate('2s ease-in',
      keyframes([
                  style({position:'relative', left:'85%'}),
                  style({position:'relative', left:'20%'}),
                  style({position:'relative', left:'5%'}),
      ])
)]),
```

นอกจากการสร้างคีย์เฟรมแบ่งสไตล์ที่ไม่เหมือนกันแต่ละการเคลื่นอที่แล้ว จากตัวอย่างที่แล้ว ทุกคีย์เฟรมใช้เวลาเท่ากันแต่ระยะการเคลื่อนไม่เท่ากันทำให้ความเร็วในการเคลื่อนไม่เท่ากันด้วย ยังมีอีกเวลาที่กำหนดระยะเวลาคีย์เฟรมให้ไม่เท่ากันได้ด้วยคำสั่ง **offset**

คำสั่ง offset ใช้กำหนดเวลาที่เคลื่อนในแต่ละคีย์เฟรม โดยเวลารวมคือ หนึ่ง หรือ 100% ของเวลารวมทั้งหมด เราสามารถกำหนดเวลาแต่ละการเคลื่อนที่ของแต่ละคีย์เฟรมให้ต่างกันได้ เช่น ในตัวอย่างต่อไปนี้ให้คีย์เฟรมแรกใช้เวลาทั้งหมด 0 เป็นจุดเริ่มต้น คีย์เฟรมที่สองใช้เวลา 0.1 หรือ 10% ของเวลาทั้งหมด และที่คีย์เฟรมสุดท้ายใช้เวลาที่สุดท้ายคือ 1

Code 10. src/app/home/home.component.ts

```
transition('start=>end', [ animate('2s ease-in',
      keyframes([
                  style({position:'relative', left:'5%', offset: 0}),
                  style({position:'relative', left:'70%', offset: 0.1}),
                  style({position:'relative', left:'85%', offset: 1.0}),
      ])
)]),
```

จากการตัวเวลาในแต่ละคีย์เฟรมใหม่นี้ การเคลื่อนที่แรก จากระยะ 5% ไปที่ระยะ 70% ใช้เวลาน้อยมากจึงทำให้เคลื่อนที่เร็ว และเมื่อไปที่คีย์เฟรมสุดท้ายมีระยะนิดเดียวแต่ใช้เวลามาก ทำให้เคลื่อนที่ช้าลงมาก

ลำดับชั้นของการทำเคลื่อนไหว

ที่ผ่านมาเราทำให้มีการเคลื่อนไหวผ่าน trigger สองตัวแล้ว ซึ่งแยกทำงานอิสระต่อกัน ยังมีการจัดการเคลื่อนไหวให้มีลำดับชัน โดยมี trigger เดียวแต่มีสถานะ (state) สองตัว หรือหลายตัว อยู่ในลำดับชันของ HTML ที่ต่างกัน การทให้ชั้นในมีการเคลื่อนไหวในลำดับที่ต่างกันได้ใช้ฟังก์ชัน query( ) อ่านค่าส่วนที่ต้องการควบคุม (element, class, id, view) และถ้าต้องการหน่วงเวลาของแต่ส่วนควบคุมจะใช้ฟังก์ชัน stagger( ) ดังนั้นต้องนำเข้าฟังก์ชันเหล่านี้ด้วย

```
import {  query, stagger  } from '@angular/animations'
```

วิธีการควบคุมคือ ตัวนอกต้องสืบค้นส่วนประกอบภายในที่ต้องการให้มีการเคลื่อนไหวด้วยฟังก์ชัน query( ) และสั่งให้เคลื่อนไหวด้วยฟังก์ชัน transition( ) ให้มีการเคลื่อนไหวตามลำดับที่ต้องการ

ตัวอย่างต่อไปนี้ใช้ทรงกลมสองวง วงนอกสีน้ำเงิน ส่วนวงในสีแดง ต้องการให้เมื่อเริ่มแสดงผล วงกลมนอกซึ่งเป็นคลาส outter เคลื่อนไหวจากสีจางไปสีเข้ม ลำดับต่อมาวงกลมในซึ่งเป็นคลาส inner แสดงผลจากสีจากไปสีเข้ม

Code 11. src/app/home/home.component.html

```
<div [@outer-inner]="isShow ?'show':'hide'" (click)='toggleShow()'>
<div class='outter'
     style="background-color:blue;border-radius:50px;height:100px;width:100px;
margin:10px auto">
        
       <div class='inner'
            style="margin:0px auto;
            background-color:red;border-radius:25px;height:50px;width:50px;">
       </div>
</div>
</div>
```

จากคอนโพเน้นท์นี้ ใช้ @outer-inner กำหนดสถานะ show หรือ hide ผ่านการคลิกของฟังก์ชัน toggleShow( ) จัดสไตล์วางวงกลมทั้งสองวงอยู่กึ่งกลางของหน้าเว็บ

Code 12. src/app/home/home.component.ts

```
isShow = true;
toggleShow(){
       this.isShow = !this.isShow;
}
```

ฟังก์ชัน toggleShow( ) ทำหน้าที่สลับค่า isShow เป็น true หรือ false เพื่อ @outer-inner มีสถานะเป็น show หรือ hide

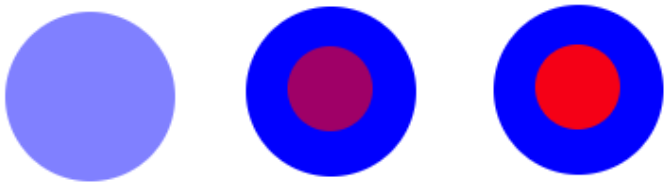
Code 13. src/app/home/home.component.ts

```
trigger('outer-inner',[
       state('show', style({opacity:1})),
       state('hide', style({opacity:0})),
       transition('*=>show',[
             query('.outter', style({ opacity: 0 })),
             query('.inner', style({ opacity: 0 })),
             query('.outter', animate(1000, style({opacity:1}))),
             query('.inner', animate(1000, style({opacity:1}))),
       ]),
       transition('*=>hide',[
             query('.outter', style({ opacity: 1 })),
             query('.inner', style({ opacity: 1})),
             //ให้ลำดับของ inner ทำงานก่อนจึงงานการเคลื่อนไหวก่อน outter
             query('.inner', animate(1000, style({opacity:0}))),
             query('.outter', animate(1000, style({opacity:0}))),
       ]),
```

```
]),
```

เมื่อเริ่มแสดงผล การเปลี่ยนสถานะเป็น show (*=>show) ฟังก์ชัน query( ) สืบค้นคลาส outter และ inner กำหนดสไตล์มีความเข้มของสีเป็น 0 ซึ่งจะเป็นจุดเริ่มต้นการเคลื่อนไหว โดยให้ คลาส outter เคลื่อนไหวก่อนให้มีสีค่อย ๆ เข้มขึ้นในหนึ่งวินาที (1000) ต่อด้วยคลาส inner เคลื่อนไหวให้มีสีค่อย ๆ เข้มขึ้นในหนึ่งวินาทีเช่นกัน

เมื่อมีการคลิกวงกลม ผ่านฟังก์ชัน toggleShow( ) สถานะจะเปลี่ยนเป็น hide ทำให้เกิดการเปลี่ยนแปลงอีกครั้ง (*=>hide) ใช้ฟังก์ชัน query( ) สืบค้นคลาส outter และ inner กำหนดสไตล์มีความเข้มของสีเป็น 1 ซึ่งจะเป็นจุดเริ่มต้นการเคลื่อนไหว โดยให้ คลาส inner เคลื่อนไหวก่อนให้มีสีค่อย ๆ จางลงจนหายไปในหนึ่งวินาที (1000) ต่อด้วยคลาส outter เคลื่อนไหวให้มีสีค่อย ๆ จางลงจนหายไปในหนึ่งวินาทีเช่นกัน



รูป 4 ระดับความเข้มในระยะต่าง ๆ

ยังมีการเคลื่อนไหวเป็นลำดับขั้นอีกวิธีหนึ่งคือ การกำหนดเวลาหน่วงที่ต่างกันด้วยฟังก์ชัน stagger( ) ในตัวอย่างที่ผ่านมาเราเคยใช้ตารางรายชื่อ ในการคลิกให้ให้แถวหายไปด้วยใช้ฟังก์ชัน disappear(i) ให้นำตารางนั้นมาปรับปรุง โดยให้ใส่ @showHide ที่หน้า <table> แทนการสร้างใน <tr> ดังตัวอย่างต่อไปนี้

Code 14.src/app/home/home.component.html

```
<table @showHide class='table'>
<tr    *ngFor ="let v of users; index as i"
       (click)='disappear(i)'>
       <td>{{v}}</td></tr>
</table>
```

ต่อมาแก้ไขการเคลื่อนไหวใหม่ ให้มีการหน่วงเวลา 300 หรือ 0.3 วินาที โดยเพิ่มฟังก์ชัน stagger( ) สำหรับตัวเลข 300 จะส่งผลให้การหน่วงเวลาเดินหน้า ทำให้การแสดงแถวแรกไปจนแถวสุดท้าย แต่ถ้าใส่เป็น -300 จะทำให้การหน่วงเวลาย้อนหลัง จะทำให้แสดงผลแถวสุดท้ายก่อนไล่ไปแถวแรก ตัวอย่างโปรแกรมนี้ใช้แทน transition เดิมของทริกเกอร์ showHide

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

```
transition(':enter', [
       query('tr', style({ opacity: 0 })),
       query('tr', stagger(300, [animate(1000, style({ opacity: 1 }))])),
]),
```

รูป 5 แสดงการหน่วงเวลา ณ ขณะเริ่มต้น (ซ้าย) และเมื่อเสร็จสิ้นการแสดงผล (ขวา)

เคลื่อนไหวเป็นคู่ขนาด และตามลำดับ

ที่ผ่านมาเราได้ทำการเคลื่อนไหวภายฟังก์ชัน animation( ) ซึ่งใส่สไตล์การเคลื่อนไหวได้หลายแบบภายใต้ฟังก์ชันนี้ ทำให้เคลื่อนไหวทำงานพร้อม ๆ กัน หรือคู่ขนาดกัน แต่ถ้าต้องการให้เวลาหรือรูปแบบเวลาของแต่ละสไตล์ต่างกัน เราต้องกำหนดฟังก์ชัน animation( ) ขึ้นมาใหม่ แต่อย่างไรก็ตามเราสามารถทำให้แต่ละการเคลื่อนไหวของฟังก์ชันนี้ทำงานพร้อมกันได้ด้วยฟังก์ชันการจับรวม คือ group( ) หรือทำให้ละการเคลื่อนไหวทำงานตามลำดับได้ด้วยฟังก์ชัน sequence( ) และต้องนำเข้าสองฟังก์ชันนี้ด้วย

```
import {  group, sequence } from '@angular/animations';
```

ตัวอย่างต่อไปนี้แสดงการทำงานคู่ขนาดภายใต้ฟังก์ชัน group( ) ของแต่ละการเคลื่อนไหวของฟังก์ชัน animation( ) โดยฟังก์ขัน animation( ) แรกทำให้เกิดการเคลื่อนไหวจากซ้ายสุดไปขวาสุด (translateX(0)) และมีความเข้มของสีเพิ่มจนได้ 0.2 ทั้งหมดใช้เวลา 3 วินาที ในขณะเดียวกันก็มีอีกความเคลื่อนไหวหนึ่งเพิ่มความเข้มจนเต็มร้อยเปอร์เซ็นต์ แต่ทำเสร็จที่หลังไปอีก 3 วินาที (เริ่มพร้อมกันแต่ช้ากว่า 3 วินาที)

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

```
trigger('flyOut', [
    state('in', style({ width: "100%" })),
    transition('void => *', [
      style({ width: 0, transform: 'translateX(0)', opacity: 0.2 }),
      group([
        animate('3s ease', style({
          transform: 'translateX(0)',
          width: "100%",
        })),
        animate('6s ease', style({
          opacity: 1
        }))
      ])
    ]),
]),
```

สำหรับตัวทดสอบการเคลื่อนไหวใช้กล่องผืนผ้า สีเทา เพื่อทดสอบแล้วเมื่อแรกเริ่ม กล่องนี้จะเคลื่อนไหวตามที่กำหนดในทริกเกอร์ flyout

Code 17.src/app/home/home.component.html

```
<div  [@flyOut]="'in'"
style="width:100%; height:100px; background-color:gray">
</div>
```

สำหรับการเคลื่อนไหวตามลำดับ เราทดสอบได้เพียงแก้จาก group( ) ไปเป็น sequence( ) ซึ่งทำตามลำดับเคลื่อนไหวที่ระบุในฟังก์ชันนี้

สร้างเป็นไฟล์เก็บเฉพาะการเคลื่อนไหว

การเคลื่อนไหวที่สร้างไว้ยังสามารถนำไปสร้างเป็นไฟล์ เพื่อให้คอมโพเน้นท์อื่น ๆ นำไปใช้ซ้ำได้ เช่น สร้างการเคลื่อนไหวในไฟล์ชื่อ animations.ts

Code 18. animations.ts

```
import { style, animate } from '@angular/animations';

export const animation1 =  animate('3s ease',
   style({transform: 'translateX(0)', width: "100%", })
)
export const animation2 =  animate('3s ease',
   style({transform: 'translateX(0)', width: "100%", })
)
```

ส่วนการนำไปใช้ซ้ำก็เพียงนำเข้าของคอมโพเน้นท์ ให้นำเข้าตามชื่อตัวแปรที่แทนความเคลื่อนไหว เช่น ต้องการนำเข้าสองการเคลื่อนไหว animation1 และ animation2

```
import { animation1, animation2 } from './animations';
```

เมื่อนำเข้าแล้ว เราสามารถใช้ชื่อการเคลื่อนไหว แทนความเคลื่อนไหว ดังตัวอย่างต่อไปนี้

Code 19. src/app/home/home.component.ts

```
trigger('flyOut', [
    state('in', style({ width: "100%"  })),
    transition('void => *', [
      style({ width: 0, transform: 'translateX(0)', opacity: .2 }),
      sequence([ animation1, animation2 ])
    ]),
])
```

หยุดการเคลื่อนไหว

ในการหยุดการเคลื่อนไหวของ transition ต่าง ๆ ให้ใช้ @.disabled เป็นตัวแปรเข้าของ @HostBinding( ) จึงต้องนำเข้าส่วนนี้ก่อน แล้วใส่กับคอมโพเน้นท์หลัก

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

ดังตัวอย่างต่อไปนี้ ได้สร้างตัวแปร animationsDisabled เป็น false เพื่อตั้งเป็นค่าเริ่มต้น โดยการเปลี่ยนแปลงค่าตัวแปรนี้ขึ้นอยู่กับฟังก์ชัน toggleAnimation( ) ที่รองรับการคลิกจากหน้า HTML แล้วจะสลับค่า true และ fasle

Code 20. src/app/home/home.component.ts

```
@HostBinding('@.disabled')
public animationsDisabled = false;

toggleAnimations() {
    this.animationsDisabled = !this.animationsDisabled;
}
```

สำหรับหน้า HTML ให้ผูกข้อมูล animationsDisabled ไว้กับ เช็กบ็อก (checkbox) โดยมีฟังก์ชัน toggleAnimation( ) รองรับการคลิก

Code 21. src/app/home/home.component.ts

```
Toggle All Disable or Enable The animations <input type="checkbox"
[checked]="!animationsDisabled" (click)="toggleAnimations()"/>
```

เมื่อทดสอบคลิกไปมาบนเช็กบ็อกนี้จะทำให้การเคลื่อนไหวที่กำหนดใน transition ต่าง ๆ หยุดทำงานทั้งหมด แต่การเปลี่ยนแปลง ตาม CSS ยังคงเดิม เช่น การคลิกแล้วหาย การคลิกแล้วย้ายยังทำงานเหมือนเดิม แต่การย้ายหรือเคลื่อนตามเวลาจะไม่ทำงาน

การเคลื่อนไหวตามเส้นทาง

เมื่อเปลี่ยนเส้นทาง หรือเปลี่ยนหน้าเว็บหนึ่งไปยังอีกหน้าเว็บหนึ่ง หรือเปลี่ยนคอมโพเน้นท์แสดงผลตามทำกำหนดไว้ในโมดูล app-routing การสร้างการเคลื่อนไหวเมื่อเปลี่ยนเส้นทางก็อีกวิธีหนึ่งที่ทำได้

เพื่อทดสอบการเคลื่อนไหว ให้สร้างคอมโพเน้นท์เพิ่มอีกสองคอมโพเน้นท์

```
ng generate component courses
ng generate component users
```

และต้องเพิ่มเส้นทางของสองคอมโพเน้นท์นี้ไปยัง app-routing.module.ts นอกจากเพิ่มเส้นทางแล้วยังกำหนดการเคลื่อนไหวในเส้นทางด้วย

Code 22. src/app/app-routing.module.ts

```
const routes: Routes = [
  { path:'home', component: HomeComponent},
  { path:'courses', component: CoursesComponent, data:{animation:'CoursesPage'} },
  { path:'users', component: UsersComponent, data:{animation: 'UsersPage'}},
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path:'**', component: PageNotFoundComponent },
];
```

จากตัวอย่างนี้จะเห็นการกำหนดการเคลื่อนไหวไว้ในเส้นทาง ด้วยเพิ่มส่วน data:{ } ในส่วนปีกกา จะเป็นข้อมูลที่สร้างหน้าเว็บหลัก (AppComponent) โดยมีคีย์เป็น animation และข้อมูลเป็นชื่อสถานะ (CoursesPage, UsersPage) ในการเปลี่ยนไประหว่างสถานะที่กำหนดไว้ใน transition ของ trigger

ทั้งสองคอมโพเน้นท์ ให้ใส่ข้อมูลอะไรก็ได้ในหน้า HTML เพื่อเป็นการจำลองข้อมูลแสดงผล ต่อมาตามเส้นทางใหม่นี้ให้ทำการเพิ่มตัวนำทางไปยังหน้าหลัก โดยวางบนสุดของหน้าเว็บ

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

```
<nav class="nav">
  <a class="nav-link active" routerLink="/home">home</a>
  <a class="nav-link" routerLink="/courses">Courses</a>
  <a class="nav-link" routerLink="/users">Users</a>
</nav>
```

เมื่อกำหนดเส้นทางใน <nav> แล้ว สิ่งสำคัญที่ต้องแก้ไขคือ <router-outlet> ให้รับข้อมูลการเปลี่ยนเส้นทางเพื่อส่งต่อไปการเปลี่ยนสถานะ

Code 24. src/app/app.component.html

```
<div [@routeAnimations]="prepareRoute(outlet)">
  <router-outlet #outlet="outlet"></router-outlet>
</div>
```

ในตัวอย่างนี้ ใช้ฟังก์ชัน prepareRoute(outlet) เพื่ออ่านสถานการณ์เปลี่ยนสถานะของเส้นทาง แล้วส่งต่อไปยังตัวแปร @routeAnimations ซึ่งกำหนดเป็นชื่อ trigger ในการเคลื่อนไหว

สำหรับฟังก์ชัน prepareRoute( ) มีตัวแปรเข้ามีไทป์เป็น RouterOutlet จึงต้องนำเข้าไทป์นี้ด้วย

```
import { RouterOutlet } from '@angular/router';
```

ฟังก์ชันนี้ใช้การคืนค่าโดยเงื่อนไข “และ” หรือ “&&” เพื่อให้มันใจว่ามีตัวแปร out และมีข้อมูลการเคลื่อนไหว ซึ่งก็คือข้อมูลชื่อสถานะของคอมโพเน้นท์ที่ระบุในโดดูลเส้นทางนั้นเอง

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

```
prepareRoute(outlet: RouterOutlet) {
	return outlet &&
		outlet.activatedRouteData &&
		outlet.activatedRouteData.animation;
}
```

กำหนดการเคลื่อนไหวระหว่างเส้นทาง

ในการเคลื่อนไหวระหว่างเส้นทางต้องการให้เมื่อคลิกเลือกเส้นทางระหว่าง courses กับ users ซึ่งมีสถานะการเคลื่อนที่เป็น UsersPage กับ CoursesPage จะมีการเลื่อนหน้าเว็บจากซ้ายไปขวา กล่าวคือ เมื่อขณะอยู่ที่เส้นทาง courses เมื่อคลิกเปลี่ยนเส้นทางไปยัง users จะทำให้หน้าเว็บของเส้นทาง courses เคลื่อนที่จากซ้ายไปขวา และหน้าเว็บของเส้นทาง users เคลื่อนที่จากซ้ายไปขวามาแทนที่หน้าเว็บที่หายไปของเส้นทาง courses

กลไกสำคัญคือ จับกลุ่มด้วยคำสั่ง group( ) ของหน้าเว็บทั้งสองให้เคลื่อนที่พร้อม ๆ กัน โดยที่หน้าเว็บหนึ่งเคลื่อนเข้า และอีกหน้าเว็บหนึ่งเคลื่อนออก การใช้คำสั่ง query( ) เพื่ออ่านหน้าเว็บที่เคลื่อนเข้า และออกด้วย :enter, :leave ซึ่งหมายถึงจากสถานะใด ๆ ไปสู่การเข้าหรือออก ดูจากตัวอย่างต่อไปนี้

Code 26. src/app/slideAnimation.ts

```
import {
  trigger, style, animate, transition, query, group,
} from '@angular/animations'
```

```
export const slideAnimation =
  trigger('routeAnimations', [
    transition('UsersPage <=> CoursesPage', [
      style({ position: 'relative' }),
      query(':enter, :leave', [
        style({
          position: 'absolute',
          top: 0,
          left: 0,
          width: '100%'
        })
      ]),
      query(':enter', [
        style({ left: '-100%' })
      ]),
      group([
        query(':leave', [
          animate('300ms ease-out', style({ left: '100%' }))
        ]),
        query(':enter', [
          animate('300ms ease-out', style({ left: '0%' }))
        ])
      ]),
    ]),
]);
```

จากตัวอย่างนี้ขณะเริ่มต้นให้ทุกสถานะอยู่ตำแหน่งขิดซ้ายสุดด้วยความกว้างเต็มที่ (100%) ในส่วนฟังก์ชัน group( ) เป็นส่วนการเริ่มการเคลื่อนไหวระหว่างสถานะเข้า และออก โดยเลื่อนจากชิดซ้ายไปสู่ขวา (left: 0% -> 100%) นี้เป็นการเคลื่อนไหว แต่อย่างไรก็ตามให้ถถานะเข้า (:enter) ต้องการให้อยู่ซ้ายไป -100% ก่อนเข้าสู่ 0% และจากไป 100%

ในกรณีที่ต้องการให้เลื่อนจากขวามาซ้ายก็ทำเพียงเปลี่ยนคำว่า left ที่มีในตัวอย่างทั้งหมดเป็น right ซึ่งมีอยู่ 4 ที่ของทั้ง 4 ฟังก์ชัน query( )

นำไฟล์ slideAnimation.ts ไปใช้กับไฟล์ app.component.ts ซึ่งเป็นหน้าหลัก โดยการนำเข้า slideAnimation ดังตัวอย่างต่อไปนี้

Code 27. src/app/app.component.ts

```
import { slideAnimation } from './slideAnimation';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  animations: [
    slideAnimation
  ]
})
```

สรุป

มาถึงตรงนี้ก็เห็นการเคลื่อนที่ในลักษณะต่างๆ การมีความรู้เรื่อง CSS ของ HTML 5 ซึ่งมีคุณสมบัติในการเคลื่อนที่อยู่แล้ว ได้นำมาประยุกต์ใช้กับฟังก์ชันต่าง ๆ ของแองกูล่าร์ ดังสรุปการใช้งานเป็นตารางดังนี้

ตาราง 3 สรุป API ของการสร้างความเคลื่อนที่

ฟังก์ชัน	การใช้งาน
trigger( )	เริ่มทำการเคลื่อนที่ โดยผู้การเคลื่อนที่ไว้กับ HTML ตามชื่อชอง trigger ใช้รูปแบบการผูกแบบอาฐูรย์
style( )	กำหนด CSS ในการเคลื่อนที่ ใช้รูปแบบการประกาศแบบ ออบเจ็กต์ ({ })
state( )	กำหนดชื่อสถานะ และกำหนด style( )
animate( )	กำหนดเวลาที่ใช้ในการเคลื่อนที่
transition( )	กำหนดลำดับการเคลื่อนที่ระหว่างสองสถานะ และกำหนด animate( )
keyframes( )	กำหนดลำดับการเปลี่ยนแปลงของ style( ) ภายใต้ animate( )
group( )	กำหนดกลุ่มการเคลื่อนที่ ให้ทำงานแบบคู่ขนาน
query( )	ใช้ค้นหาอิลิเม้นท์ของ HTML หร้อทั้งหน้าผ่านสถานะ ภายใต้ อิลิเม้นท์ปัจจุบัน
sequence( )	ระบุลำดับการเคลื่อนที่
stagger( )	การหน่วงเวลา ในแต่ละเคลื่อนไหว ใช้คู่กับ query( )
animation( )	การเคลื่อนไหว

คำถามทบทวน

1. ชื่อที่ใช้อ้างอิงในการการเคลื่อนไหวของไฟล์ HTML เป็นชื่ออะไร
2. เวลาในการเคลื่อนไหว ถ้าระบุหน่วยเวลา จะถือเป็นหน่วยเวลาใด
3. สถานะใด ๆ เขียนแทนด้วยสัญลักษณ์ใด
4. สถานะ void หมายถึงการเวลาทำงานในช่วงใด
5. มีเหตุการณ์ของการเคลื่อนไหวใดบ้างที่ตรวจจับได้
6. เวลาในคีย์เฟรมมีค่าเป็นอย่างไร
7. การใช้ query( ) ใช้เพื่อการใด
8. การหน่วงเวลาใช้ฟังก์ชันใด
9. การสร้างการเคลื่อนไหวในหลายส่วน ควบคุมได้ด้วยฟังก์ชันใด
10. การใส่การเคลื่อไหวในเส้นทาง กำหนดได้อย่างไร

แบบฝึกหัด

1. ในรายการนำทาง ที่ได้สร้างมีสองรายการคือ Courses กับ Users ซึ่งเมื่อมีการคลิกแล้วจะมีเนื้อหาแต่ละหน้าเคลื่อนไหวจากซ้ายไปขวาทั้งสองเนื้อหา ให้ปรับปรุง แก้ไขคือ เมื่อขณะหน้าปัจจุบันอยู่ที่ Users แต่ผู้ใช้คลิกที่ Courses ให้เนื้อหาของ Courses เลื่อนจาก ขวามาซ้าย