**26. Angular Animations**

403. Introduction

**Angular Animations Note:**

Making Animations Work with Angular 4+

With the release of **Angular 4**, the general syntax of Angular Animations didn't change.

However, the animation functions were moved into their own package and you now also need to add a special module to your imports[] array in the **AppModule**.

Specifically, the following adjustments are required:

1. You probably need to install the new animations package (running the command never hurts): npm install --save @angular/animations

2. Add the BrowserAnimationsModule to your imports[] array in **AppModule**

3. This Module needs to be imported from @angular/platform-browser/animations' => import { BrowserAnimationsModule } from '@angular/platform-browser/animations' (in the AppModule!)

4. You then import trigger , state , style etc from @angular/animations **instead of**@angular/core

That's all!

**26. Angular Animations**

405. Animations Triggers and State

**Angular Animations Setup 1.1:**

ანიმაციები კომპონენტში თავად **@Component** დეკორატორში დეკლარირდებიან, ანუ ემატება ახალი ფროფერთი, რომელიც მასივს წარმოადგენს, რომელშიც ემატება **trigger**(იგი ყველა ანიმაციას გააჩნია), რომელიც უნდა დაიმპორტდეს **@angular/animations**-დან.

იგი წარმოადგენს ფუნქციას: **triger()**. პირველ არგუმენტად ჩაეწოდება თავად ტრიგერის სახელწოდება (რაც გვსურს), რომელიც **html**კომპონენტში შეგვიძლია ფროფერთი ბაინდინგის მსგავსად ელემენტს მივაბათ და გავუტოლოთ ფროფერთის, რომელიც ts-ში იქნება დეკლარირებული (მაგ. **[@divState]="state"** ), ხოლო მეორე არგუმენტად მასივი, რომელშიც რამდენიმე state() მეთოდი შეიძლება გამოვიყენოთ (state() იმპორტდება **@angular/animations**-დან).

თავად **ts**ფაილში კი შევქმნათ ფროფერთი (რომელიც ზემოთ დავაკავშრეთ) შემდეგნაირად:

**state = "state"**

აღნიშნული მნიშვნელობები სრულიად ჩვენზეა დამოკიდებული, უბრალოდ, როდესაც **state()** მეთოდში გამოვიძახებთ, იგივე სახელი უნდა ერქვათ.

**26. Angular Animations**

405. Animations Triggers and State

**Angular Animations Setup 1.2:**

ასევე თავად სტეიტში მეორე არგუმენტად ვიძახებთ style() მეთოდს და შიგნით **css**ჩაიწერება (ობიექტად).

შეგვიძლია ორი სტეიტი შევქმნათ და თითოეულს ჩავაწოდოთ სტილები. თუმცა ანიმაცია ჯერ არ იმუშავებს.

მის ასამუშავებლად შემდეგში ვიმუშავებთ

თუმცა, **ts**კოდი ახლა ასე გამოიყურება:

1. @Component({
2. selector: 'app-root',
3. templateUrl: './app.component.html',
4. animations: [
5. trigger('divState', [
6. state('normal', style({
7. 'background-color': 'red',
8. transform: 'translateX(0)'
9. })),
10. state('highlight', style({
11. backgroundColor: 'blue',
12. transform: 'translateX(100px)'
13. }))
14. ])
15. ]
16. })
17. export class AppComponent {
18. state = 'normal'
19. }

**26. Angular Animations**

406. Switching between States

**Angular Animations Setup 1.3:**

შემდეგ შევქმნათ მეთოდი, რომელიც **state**ფროფერთის მნიშვნელობებს შეუცვლის:

1. onAnimate(){
2. this.state == 'normal' ? this.state = 'highlight' : this.state = 'normal'
3. }

აღნიშნული მეთოდი ღილაკს მიებმება და ჩვეულებრივ ამუშავდება.

*თუ იბაგება,****css****-ის ობიექტში ერთი და იგივე სტიპი უნდა გამოვიყენოთ.*

**html:**

1. <button class="btn btn-primary" (click)="onAnimate()">Animate!</button>
2. <hr>
3. <div
4. style="width: 100px; height: 100px"
5. [@divState]="state"></div>
6. </div>

**26. Angular Animations**

407. Transitions

**Angular Animations Setup 1.4:**

ანიმაციის ვიზუალურად დასანახად, საჭიროა **transition**დავადოთ.

ამისთვის, ჩვენს**trigger()** მეთოდში ჩავამატოთ მეთოდი **transition()** (ასევე იგივენაირად იმპორტდება). შიგნით მიდის პირველ არგუმენტად, სტეიტების გადასვლა, ხოლო მეორედ animate() მეთდი (რომელიც ასევე იმპორტდება), რომელიც იღებს მილიწამებს არგუმენტად:

1. transition('normal => highlight', animate(300))

კოდ იახლა ასე გამოიყურება:

1. animations: [
2. trigger('divState', [
3. state('normal', style({
4. 'background-color': 'red',
5. transform: 'translateX(0)'
6. })),
7. state('highlight', style({
8. backgroundColor: 'blue',
9. transform: 'translateX(100px)'
10. })),
11. transition('normal => highlight', animate(300)),
12. transition('highlight => normal', animate(800))
13. ])
14. ]

**26. Angular Animations**

408. Advanced Transitions

**Angular Animations Advanced 1.1:**

ამის ნაცვლად:

1. transition('normal => highlight', animate(300)),
2. transition('highlight => normal', animate(800))

შეგვიძლია უბრალოდ:

1. transition('normal <=> highlight', animate(300))

ასევე, ისეთ სტილის მისანიჭებლად, რომელიც არა საწყისში ან ბოლოში, არამედ სადღაც შუაში გამოჩნდება, შეგვიძლია ცალკე ფროფერთი შევქმნათ, ასევე ცალკე დივი, სადაც მას გავტესტავთ, და ასევე ახალი**state()**მეთოდი დავამატოთ შესაბამისად.

**26. Angular Animations**

408. Advanced Transitions

**Angular Animations Advanced 1.2:**

**trigger**-ში ახალი სტეიტი:

1. state('shrunken', style({
2. backgroundColor: 'green',
3. transform: 'translateX(0) scale(0.5)'
4. })),

ასევე ახალი **transition**-ი:

1. transition('shrunken <=> \*', animate(500))

**\*** - აღნიშნული **wild**-ია, რაც ზუსტად არ განსაზღვრავს რომელ სტეიტში უნდა გადავიდეს, ანუ ნებისმიერი შეიძლება იყოს.

ასევე თავად class-ის body-ში ემატება:

1. wildState = 'normal';
2. onShrink(){
3. this.wildState = 'shrunken'
4. }

და თავად **onAnimate()** მეთოდში:

1. this.wildState == 'normal' ? this.wildState = 'highlight' : this.wildState = 'normal';

**26. Angular Animations**

409. Transition Phases

**Angular Animations Transition ფაზები:**

**transition()** მეთოდს თავადვე შეგვიძლია ჩავაწოდოთ მეორე არგუმენტად მასივი, რომელიც ასევე მიიღებს **style()**მეთოდსაც:

1. transition('shrunken <=> \*', [
2. style({
3. 'background-color': 'orange'
4. }),
5. animate(1000, style({
6. 'border-radius': '50px'
7. })),
8. animate(500)
9. ])

**26. Angular Animations**

410. The "void" State

**Angular Animation Void 1.1:**

**void**სტეიტი გამოიყენება იმ შემთხვევაში, თუ ელემენტი არ არსებობს და იქმნება ან პირიქით: transition('void => \*', animate(300)).

რამდენადაც **\***გამოვიყენეთ, **html**-ის**\*ngFor**-ის ელემენტში შემდეგნაირად ვამატებთ ამ ფროფერთის: [@list1]

აღნიშნულის ასამუშავებლად, ინიშიალ სტეიტი მაინც საჭიროა, ამიტომაც, თავად **transition()** მეთოდში მეორე არგუმენტად წავა მასივი (ანუ რა უნდა მოხდეს **transition**-ისას, სანამ საბოლოო სახეს მიიღებს):

1. trigger('list1', [
2. state('in', style({
3. opacity: 1,
4. transform: 'translateX(0)'
5. })),
6. transition('void => \*', [
7. style({
8. opacity: 0,
9. transform: 'translateX(-100px)'
10. }),
11. animate(300)
12. ]),
13. ])

**26. Angular Animations**

410. The "void" State

**Angular Animation Void 1.2:**

ასევე, პირიქითად შეგვიძლია გამოვიყენოთ, ანუ წაშლისას, სადაც ახალ **transition**-ს ჩავამატებთ:

1. transition('\* => void', [
2. animate(300, style({
3. transform: 'translateX(100px)',
4. opacity: 0
5. }))
6. ])

აქ, რამდენადაც**style()** თავად **animate()**-შია გამოყენებული, გაწერილ სტილს ბოლოს მიიღებს.

**26. Angular Animations**

411. Using Keyframes for Animations

**Angular Animation keyframes:**

ჩვენ ასევე შეგვიძლია ანიმაცია გავაკონტროლოთ დროის მიხედვით, ანუ მითითებული დროის რომელ მონაკვეთზე რა სტილი მიიღოს. **keyframes**იმპორტდება სტანრტულად, იგი ემატება **animate()** მეთოდს. **offset**ფროფერთი განსაზღვრავს ანიმაციის დროის მონაკვეთს (**0-1**) კოდი:

1. transition('void => \*', [
2. animate(1000, keyframes([
3. style({
4. transform: 'transformX(-100px)',
5. opacity: 0,
6. offset: 0
7. }),
8. style({
9. transform: 'translateX(-50px)',
10. opacity: 0.5,
11. offset: 0.3
12. }),
13. style({
14. transform: 'translateX(0px)',
15. opacity: 1,
16. offset: 1
17. })
18. ]))
19. ]),

**26. Angular Animations**

412. Grouping Transitions

**Angular Animation Group:**

ასევე შეგვიძლია სინქრონულად რამდენიმე ანიმაციის გაშვება, ანუ ერთად, ამისთვის გამოიყენება **group()** მეთოდი, რომელიც სტარტარტულად იმპორტდება:

1. transition('\* => void', [
2. group([
3. animate(300, style({
4. color: 'red',
5. })),
6. animate(800, style({
7. transform: 'translateX(100px)',
8. opacity: 0
9. }))
10. ])
11. ])

**26. Angular Animations**

413. Using Animation Callbacks

**Angular Animation Callbacks:**

მაგალითად, თუ გვსურს რაიმე კოდის გაშვება მხოლოდ მას შემდეგ, რაც ანიმაცია დაიწყება ან დასრულდება, შეგვიძლია **html**ელემენტში, სადაც ანიმაციაა გაწერილი, ჩავამატოთ შემდეგი ივენთ ლისენერი (დასაწყისის ან/და დასასრულის):

1. <div
2. style="width: 100px; height: 100px"
3. [@divState]="state"
4. (@divState.start)="animationStarted($event)"
5. (@divState.done)="animationEnded($event)"
6. ></div>

ხოლო **ts**ფაილში აღნიშნულ მეთოდებს გამოვიძახებთ და განვახორციელებთ რამეს *($event სავალდებულო არაა)*.