12 Angular Animations (26)

https://ide.c9.io/laczor/angular2
Session_26_Animations(animations-finished)
354. Setup Animations for Angular 4+
357. Animations Triggers and State
360 Advanced Transition.
361. Animation phases
362. Void State
364. Grouping Animations
365. Listening to Animation Callbacks

354. Setup Animations for 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:

You probably need to install the new animations package (running the command never hurts): **npm install --save @angular/animations**
Add the **BrowserAnimationsModule** to your `imports[]` array in **AppModule**
This Module needs to be imported from @angular/platform-browser/animations' => **import { BrowserAnimationsModule } from '@angular/platform-browser/animations**' (in the **AppModule**!)ng
You then import trigger , state , style etc from **@angular/animations** instead of @angular/core

357. Animations Triggers and State

1. Import trigger + style + transition + animate from the @angular/animations.
2. Declare the animations in the @Component decorator
3. Create a component's state variable, which is changed upon clicking
4. Assign a property binding to the html tag what we want to animate
 [@divState]="state"

app.component.ts

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

import {

  trigger,                                                     //Will Determine which html, wit

h which attribuute dcorator should we look for
```

```
  state,                                                   //Will determine the states, whi
ch styles should be implemented
  style,                                                  //Will determine the css properti
es of the actual states
  transition,                                            //transition('from state=> to stat
e', animate(800))
                                                        // back and forth animations transi
tion('from stat <=> to state', animate(800))
  animate,
  keyframes,
  group
} from '@angular/animations';


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  animations: [
    trigger('divState', [
      state('normal', style({
        'background-color': 'red',
        transform: 'translateX(0)'
      })),
      state('highlighted', style({
        'background-color': 'blue',
        transform: 'translateX(100px)'
      })),

      transition('normal <=> highlighted', animate(300)),
      // transition('highlighted => normal', animate(800))
    ])
  ]
})
export class AppComponent {
    state = 'normal';

  //Switching between states!

    onAnimate() {

      this.state == 'normal' ? this.state = 'highlighted' : this.state = 'normal';

    }


  }
```

app.component.html

```
  <button class="btn btn-primary" (click)="onAnimate()">Animate!</button>



        <div

          style="width: 100px; height: 100px"

          [@divState]="state"

        </div>
```

360 Advanced Transition.

Transitions operators:

state1 =>state2	from one to an other state
state1 <=state2	
State1 <=> * states	from to every state using wildcard *
states *<=> State1	

1. Create new triggers in the @Component decorator at **app.component.ts**
- with scaling we change the sizes
- with adding animations we can make additinal transformation

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

  import {

    trigger,

    state,

    style,

    transition,

    animate,

    keyframes,
  group
} from '@angular/animations';


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  animations: [
    trigger('divState', [
      state('normal', style({
        'background-color': 'red',
        transform: 'translateX(0)'
      })),
      state('highlighted', style({
        'background-color': 'blue',
        transform: 'translateX(100px)'
      })),
      transition('normal <=> highlighted', animate(300)),
      // transition('highlighted => normal', animate(800))
    ]),
    trigger('wildState', [
      state('normal', style({
        'background-color': 'red',
        transform: 'translateX(0) scale(1)'
      })),
      state('highlighted', style({
        'background-color': 'blue',
        transform: 'translateX(100px) scale(1)'
      })),
      state('shrunken', style({
        'background-color': 'green',
        transform: 'translateX(0) scale(0.5)'
      })),
      transition('normal => highlighted', animate(300)),

      transition('highlighted => normal', animate(800)),

      transition('shrunken <=> *', [

        style({

          'background-color': 'orange'

        }),

        animate(1000, style({

          borderRadius: '50px'

        })),

        animate(500)

      ])

    ]),

  ]

})

export class AppComponent {

  state = 'normal';

  wildState = 'normal';

//Switching between states

  onAnimate() {

    this.state == 'normal' ? this.state = 'highlighted' : this.state = 'normal';

    this.wildState == 'normal' ? this.wildState = 'highlighted' : this.wildState = 'normal';

  }


  onShrink() {

    this.wildState = 'shrunken';

  }


}
```

2. add the new states to the **app.component.html**

```
      <button class="btn btn-primary" (click)="onAnimate()">Animate!</button>

      <button class="btn btn-primary" (click)="onShrink()">Shrink!</button>
```

```
        <hr>

        <div

        style="width: 100px; height: 100px"

          [@divState]="state"

        >

        </div>

        <br>

        <div

          style="width: 100px; height: 100px"

          [@wildState]="wildState">

        </div>
```

361. Animation phases

- We can style begining and ending states during the transition

```
         transition('shrunken <=> *', [

          style({

            'background-color': 'orange'

          }),

          animate(1000, style({

            borderRadius: '50px'

          })),

          animate(500)

        ])
```

362. Void State

if an element hasn't been added yet, than this is the **void** state, For example at adding a list.!

1. Create the triggering listState + the compnent array variable which will store the values + click() methods + *ngFor
Important
void => *   --> You have to add an endstate prior animating it,
* => void   --> You can add in the animation style the end state

app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  animations: [
    trigger('divState', [
      state('normal', style({      'background-color': 'red',     transform: 'translateX(0)'     })),
      state('highlighted', style({       'background-color': 'blue',       transform: 'translateX(100px)'     })),
      transition('normal <=> highlighted', animate(300)),
      // transition('highlighted => normal', animate(800))
    ]),
    trigger('wildState', [
      state('normal', style({      'background-color': 'red',       transform: 'translateX(0) scale(1)'     })),
      state('highlighted', style({       'background-color': 'blue',       transform: 'translateX(100px) scale(1)'     })),
      state('shrunken', style({      'background-color': 'green',       transform: 'translateX(0) scale(0.5)'    })),
      transition('normal => highlighted', animate(300)),
      transition('highlighted => normal', animate(800)),
      transition('shrunken <=> *', [
        style({        'background-color': 'orange'       }),
        animate(1000, style({         borderRadius: '50px'       })),
        animate(500)
      ])
    ]),
    trigger('list1', [
      state('in', style({ opacity: 1, transform: 'translateX(0)'   })),
      //Adding, since we want to shance the css styl, we have to give the end css state and tigg
er the animation afterwards
      transition('void => *', [
        style({   opacity: 0,  transform: 'translateX(-100px)'  }),
        animate(300)
      ]),
      //Deleting, we just have to add the endstate in the animation triggering
      transition('* => void', [
        animate(300, style({   transform: 'translateX(100px)',   opacity: 0   }))
      ])
    ]),
  ]
})
export class AppComponent {
  state = 'normal';
  wildState = 'normal';
  list = ['Milk', 'Sugar', 'Bread'];

  onAnimate() {
    this.state == 'normal' ? this.state = 'highlighted' : this.state = 'normal';
    this.wildState == 'normal' ? this.wildState = 'highlighted' : this.wildState = 'normal';
  }
  onShrink() {    this.wildState = 'shrunken';  }
  onAdd(item) {   this.list.push(item); }
  onDelete(item) {   this.list.splice(this.list.indexOf(item), 1);  }

}
```

app.component.html
- Add the **[@list1]** property binding to the for loop!

```
 <div class="row">
    <div class="col-xs-12">
      <input type="text" #input>
      <button class="btn btn-primary" (click)="onAdd(input.value)">Add Item!</button>

      <hr>

      <ul class="list-group">

        <li

          class="list-group-item"

          (click)="onDelete(item)"

          [@list1]

          *ngFor="let item of list">

          {{ item }}

        </li>

      </ul>

    </div>

  </div>
```

363. Using Keyframes

- Keyframe is actually determining a css animations how long should it take, which should be the actions
app.component.ts

```
//So we will animeate the transition to 1 second,

//we pass different states of the keyfame

// then with offset: property, we can determine at what % of the keyframe should the certain st
ates be activated.

      transition('void => *', [

        animate(1000, keyframes([

          style({

            transform: 'translateX(-100px)',

            opacity: 0,

            offset: 0

          }),

          style({

            transform: 'translateX(-50px)',

            opacity: 0.5,
            offset: 0.3
          }),
          style({
            transform: 'translateX(-20px)',
            opacity: 1,
            offset: 0.8
          }),
          style({
            transform: 'translateX(0px)',
            opacity: 1,
            offset: 1
          })
        ]))
```

364. Grouping Animations

-When you would like to start multiple animations at the same time
- Pass an array of animate methods, what we would like to starttogether
app.component.ts

```
  transition('* => void', [
    group([
      animate(300, style({
        color: 'red'
      })),
      animate(800, style({
        transform: 'translateX(100px)',
        opacity: 0
      }))
    ])
  ])
```

365. Listening to Animation Callbacks

You can listen to animation events and execute some code upon completion
app.component.html

```
(@divState.start)="animationStarted($event)"

      (@divState.done)="animationEnded($event)">
```

app.component.ts

```
animationStarted(event) {

  console.log(event);

}


animationEnded(event) {

  console.log(event);

}
```