Tema 4 - Angular

Anexo
Angular Animations.

Desarrollo web en entorno cliente
IES Pere Maria Orts I Bosch

Índice

Angular Animations

El módulo de animaciones de Angular (BrowserAnimationsModule) permite crear animaciones con rendimiento nativo basadas en el estándar de Animaciones Web. Para empezar importamos dicho módulo en nuestro módulo de aplicación (AppModule).

```
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

@NgModule({
  imports: [
    ---
    BrowserAnimationsModule
  ],
  // ---
})
export class AppModule { }
```

Las animaciones están basadas en disparadores o triggers, que cambian de un estado a otro. Para asociar un disparador a un elemento HTML, usamos el carácter @ delante del nombre que queramos asociar.

```
<!-- Animación cuando el elemento entra/sale del DOM -->
<div @trigger *ngIf="exp">
  ---
</div>

<!-- Animación cuando el estado `someStateValue` cambia de valor -->
<div [@trigger]="someStateValue">
  ---
</div>
```

En el decorador del componente que contiene el elemento, establecemos las animaciones a aplicar al trigger correspondiente. Por ejemplo, para animar el elemento cuando aparece (se inserta el en DOM):

```
@Component({
    ---
    animations: [
        trigger('animateList', [
            transition(':enter', [ // El elemento "entra" al DOM
                style({ opacity: 0, transform: 'translateX(-100px)' }), // Valores iniciales
                animate('500ms ease-out', style({ opacity: 1, transform: 'none' })) // Animación
            ])

        ])
    ]
})
```

Esta animación hace que el elemento con el trigger '@animateList', cuando aparezca, pasará del estado invisible (opacity: 0) y desde la derecha (translate: '-100px'), a ser visible y estar en su posición correspondiente en 500ms. También existe el estado ':leave' cuando se elimina el elemento del DOM.

Transiciones entre estados

Además de "entrar" o "salir" del DOM, se pueden crear itris estados y transiciones asociadas a los mismos. Además del nombre del estado, podemos usar el estado especial (*) para indicar cualquier estado, y void para indicar que el elemento no está en el DOM. Para especificar transiciones usaremos 'estado1 => estado2'.

```
<product-item [@animateList]="prod.state" (click)="toggleSelect(prod)"
...></product-item>

toggleSelect(prod: IProduct) {
    prod.state = prod.state === 'selected' ? '' : 'selected';
}

animations: [
    trigger('animateList', [
        state('selected', style({ borderLeft: '40px lightgreen solid' })),
        transition('* => selected', animate('200ms ease-in')),
        transition('selected => *', animate('200ms ease-out')),
        ...
    ])
]
```

Hemos creado el estado selected (seleccionado) y establecido 2 nuevas transiciones:

- De cualquier estado anterior a selected (* => selected)
- De selected a cualquier otro estado (selected => *)

Animaciones controladas desde el elemento padre

Por ejemplo, para animar una lista de elementos con un retraso entre ellos, lo tiene que controlar el elemento padre de dichos elementos.

```
<div *ngIf="products.length"  @productList>
    ...
    <product-item ...>
    </product-item>
</div>
```

Usando la función query, podemos seleccionar cualquier elemento interno y animarlo. Si lo combinamos con la función stagger, se puede introducir un retraso entre la animación de cada elemento hijo.

```
@Component({
  selector: 'product-list',
  templateUrl: './product-list.component.html',
  styleUrls: ['./product-list.component.css'],
  animations: [
    ...
    trigger('productList', [
      transition(':enter', [
        query('product-item', [
          style({ opacity: 0, transform: 'translateX(-100px)' }),
          stagger(100, animate('500ms ease-out', style({ opacity: 1, transform:
'none' })))
        ])
      ])
    ])
  ]
})
```

Animando las rutas

Para animar las transiciones entre rutas, añadiremos un trigger al elemento que contiene el componente router-outlet (el padre). El estado del router nos lo devolverá un método y será un valor llamado 'animation' que asociaremos a cada ruta.

```
<div class="container-fluid" [@routeAnimation]="getState(routerOutlet)">
    <router-outlet #routerOutlet="outlet"></router-outlet>
</div>
```

Este método, llamado getState, recibirá el objeto RouterOutlet y obtendrá los datos asociados a la ruta (buscando el valor de animation, si lo hay).

```
export class AppComponent {
    getState(outlet: RouterOutlet) {
        // Returns the page animation name (or 'None' of it has no animation)
        return outlet.activatedRouteData.animation || 'None';
    }
}
```

Para que esto funcione, añadimos el dato 'animation' a cada ruta (que queramos animar).

```
export const PRODUCT_ROUTES: Route[] = [
  {
    path: '',
    component: ProductListComponent,
    data: { animation: 'productListPage' }
  },
  {
    path: ':id',
    component: ProductDetailComponent,
    data: { animation: 'productDetailPage' },
    ---
  },
  ---
];
```

Finalmente, añadimos en el componente de la aplicación, las transiciones entre rutas y definimos las animaciones. Al llamar a la función query, la pseudoclase :enter la tendrá la página de destino (la que entra), y :leave la página actual (la que sale). Con group agrupamos varias animaciones para que se ejecuten en paralelo.

```
@Component({
    ---
    animations: [
        trigger('routeAnimation', [
            transition('productList => productDetail', [
                query(':enter, :leave', style({ position: 'absolute', width: '100%' })),
                query(':enter', style({ transform: 'translateX(100%)' })),
                group([
                    query(':leave', [
                        animate('0.5s', style({ transform: 'translateX(-100%)' })),
                    ]),
                    query(':enter', [
                        animate('0.5s', style({ transform: 'none' })),
                    ]),
                ])
            ]),
            transition('productDetail => productList', [
                query(':enter, :leave', style({ position: 'absolute', width: '100%' })),
                query(':enter', style({ transform: 'translateX(-100%)' })),
                group([
                    query(':leave', [
                        animate('0.5s', style({ transform: 'translateX(100%)' })),
                    ]),
```

```
                query(':enter', [
                    animate('0.5s', style({ transform: 'none' })),
                ]),
            ])
        ])
    ])
})
export class AppComponent {
    ...
}
```

Más ejemplos e información:

http://animationsftw.in

https://angular.io/guide/animations

https://angular.io/guide/transition-and-triggers

https://medium.com/codingthesmartway-com-blog/angular-6-animations-from-scratch-76e110cba5fb