Exercise (Instructions): Angular Animations Part 2

Objectives and Outcomes

In this exercise you will continue to use the animation support available in the Angular framework to add a few new features to your Angular application. At the end of this exercise you will be able to:

- Define new animations using the support available in the Angular framework
- Apply the animations to the views within your components

Refactoring the Code

- If we use multiple animations in our application, it's better to organize the code better. So we will refactor our code.
- Create a new folder named animations in the app folder, and create a file named app.animation.ts in that folder.
- Add the following code to app.animation.ts to create a new factory function that supplies the animation:

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

export function visibility() {
    return trigger('visibility', [
        state('shown', style({
            transform: 'scale(1.0)',
            opacity: 1
        })),
        state('hidden', style({
            transform: 'scale(0.5)',
            opacity: 0
        })),
        transition('* => *', animate('0.5s ease-in-out'))
    ]);
}
```

- Update dishdetail.component.ts as follows to make use of the animation factory function defined above:

```
. . .
import { visibility } from '../animations/app.animation';

. . .

  animations: [
    visibility()
  ]
  . . .
```

- Remove the following line from dishdetail.component.ts since it's already included in app.animation.ts:

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

Adding Animation Support for Route Changes

- Open app.animation.ts and add the following to it to include a new factory:

```
export function flyInOut() {
    return trigger('flyInOut',[
        state('*', style({ opacity: 1, transform: 'translateX(0)'})),
        transition(':enter', [
            style({ transform: 'translateX(-100%)', opacity:0 }),
            animate('500ms ease-in')
        ]),
        transition(':leave', [
            animate('500ms ease-out', style({ transform: 'translateX(100%)',
              opacity: 0}))
        ])
    ]);
}
```

- Import the flyInOut into menu.component.ts and then define a new animation trigger within the Component decorator in menu.component.ts to introduce a view transition when the menu component view is routed to in the application:

```
. . .
import { flyInOut } from '../animations/app.animation';

  host: {
  '[@flyInOut]': 'true',
  'style': 'display: block;'
  },
  animations: [
    flyInOut()
  ]
  . . .
```

- Apply the same updates to home.component.ts, about.component.ts and contact.component.ts
- Import flyInOut and then add the host property and the flyInOut() also to dishdetail.component.ts.

Animation to Render View from Fetched Data

- Open app.animation.ts and add the following to it to include a new factory:

```
export function expand() {
    return trigger('expand', [
        state('*', style({ opacity: 1, transform: 'translateX(0)' })),
        transition(':enter', [
            style({ transform: 'translateY(-50%)', opacity:0 }),
            animate('200ms ease-in', style({ opacity: 1, transform: 'translateX
              (0)' }))
        ])
    ]);
}
```

- Furthermore import the new function and add the following to the animations in menu.component.ts, about.component.ts, dishdetail.component.ts and home.component.ts:

```
. . .
  animations: [
    expand()
  ]
    . . .
```

- Then apply the [@expand] attribute to all those elements within the views of the above components where the data is being fetched from the service before rendering the view.
- Save all the changes and do a Git commit with the message "Animations Part 2".

Conclusions

In this exercise we learnt more about Angular animations and how they can be applied while entering and leaving views.

Mark as completed