Lab 1 - Angular Animations

In this lab you will get hands-on experience setting up Angular animations and applying them to route navigation.

The lab will cover the following tasks:

- Setting up the project
- Adding Browser Support
- Add Animation Support to Components

Part 1 - Get Started

We will now create a new Angular application.

__1. Open a command prompt window.

__2. Go to the **C:\LabWork** folder.

__3. Run this command to create a new project called **animation**.

```
ng new animation --minimal

Choose Yes for routing
Choose CSS for styles
```

Part 2 - Setup Starter Files

To save time basic parts of the application have already been implemented. In this lab we will concentrate on animating the transition between routes. To support this we will install files to create a basic application with routes. Then in later steps we will add animations that occur on the transition between routes.

__1. Copy files as directed below:

```
copy contents of: C:\LabFiles\animation\app
into this dir: C:\LabWork\animation\src\app
```

Choose Yes to overwrite files in the destination directory.

__2. Copy the following file as directed below:

```
copy this file: C:\LabFiles\animation\styles.css
into this dir: C:\LabWork\animation\src\
```

Choose Yes to overwrite the file in the destination directory.

__3. Open a command prompt and go to the root folder of the project:

```
C:\LabWork\animation
```

__4. Run the following command to compile TypeScript and start the server:

```
npm run start
```

__5. Open a browser window to the following address:

```
http://localhost:4200/
```

__6. The browser window should appear like this:

Lab setup is now complete.

Part 3 - Add Browser Support

Angular animation acts as a wrapper around Web Animations. But not all browsers currently support Web Animations. Firefox and Chrome support it but IE, Edge and Safari require an additional library to support it. In this part of the lab we will check to make sure the project includes the required library and then check some of the application files to examine the support required for animations.

__1. Open another command prompt and navigate to the root of the project:

```
C:\LabWork\animation
```

__2. Run the following command to install the web animations package:

```
npm install web-animations-js@2.2.2 --save
```

This will install the web-animations package to the node_modules directory and add it to the package.json file.

__3. Use file manager to check for the library at the following location:

```
{project root}\node_modules\web-animations-js\web-animations.min.js
```

__4. Open the project's index.html in your text editor. The file can be found in the project's 'src' directory.

__5. Edit the **C:\LabWork\animation\src\polyfills.ts** file and add the line shown below at the bottom of the file:

```
import 'web-animations-js';
```

Note: in this course we will be using the Chrome browser which includes built-in support for animations. The use of the 'web-animations-js' package is shown here just in case you need to use it in a project outside the lab where the client browser is one of the ones that requires it.

__6. Save the file.

__7. Go to the command prompt and restart the application:

```
Ctrl-C
npm run start
```

__8. Run the application in your browser (http://localhost:4200).

__9. Open the Developer Tools and confirm that there are no errors in the JavaScript console.

With these libraries and settings in place we are ready to start working with Angular Animations.

Part 4 - Create an Animations.ts file

In this lab we will add animations so that each component seems to slide in from the left when it is brought up and slide out to the right when it is dismissed. To do this we need to add code that defines these transitions to each component. Since we want all the components to act the same way the code for the transitions in each component will also be the same. Instead of cutting and pasting the same code into all the components we are going to define the animation information in its own file and export it so it can be reused with each component.

__1. Open the following file in your text editor.

```
\src\app\app.module.ts
```

__2. Add the following new import after the other imports near the top of the file. This should all be on one line even though it wraps below:

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

__3. Modify the 'imports' array to include the newly added 'BrowserAnimationsModule'. Make sure to add the 's' on 'Animations' in the module name and the comma after the new entry.

```
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule,
    BrowserAnimationsModule,
    AppRoutingModule
  ],
```

__4. Save and close the file.

__5. Create a new file in your text editor. Save it with the following name and location in the project. Make sure the file extension is correct and the file isn't saved with a .txt extension.

```
\src\app\animations.ts
```

__6. With the file still open in the text editor enter the following text into the file and re-save it. If you like you can copy the contents from **C:\LabFiles\animations.ts** instead.

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

export class Animations {
static styleStart = style({transform:'translateX(-100%)', opacity: 0 });
static styleCurrent = style({ transform: 'translateX(0)', opacity: 1 });
static styleEnd = style({ transform: 'translateX(100%)', opacity: 0 });

 static page  =
     trigger('routeAnimation', [
              state('*', Animations.styleCurrent ),
              transition('void => *', [ Animations.styleStart,
                animate('0.5s ease-in-out') ] ),
              transition('* => void', animate('0.5s ease-in-out',
                Animations.styleEnd))
 ]);
}
```

This file defines a class with a static member named "page" that contains the animation settings. styleStart defines a component's position when it is offscreen to the left before it slides in. styleCurrent defines a component's position when it has slid in from the left and is then visible on screen. styleEnd defines a component's position after it slides out to the right. The "page" property is defined by executing the Angular Animation "trigger" function. The trigger name "routeAnimation" indicates that transitions should be invoked when the user navigates to or away from a given component.

__7. Save and close the file.

__8. Scroll up in the command prompt to check there are no compilation errors.

Part 5 - Add Animation to the Home Component

We need to import the animation settings to each component and then set up the component to use them.

__1. Open \src\app\home\home.component.ts in your text editor.

__2. Add the following import after the other imports at the top of the file:

```
import { Animations } from '../animations';
```

__3. Add Animations.page in an animations array to the @Component metadata section of the component (as shown in bold below) don't forget the comma "," between 'animations:' and the entry that comes before it:

```
@Component({
  selector: 'app-home',
  templateUrl: './app/home/home.component.html',
  styleUrls: ['./app/home/home.component.css']
  , animations: [ Animations.page ]
})
```

__4. Add the following code (in bold) above the constructor() in the HomeComponent class:

```
export class HomeComponent implements OnInit {

 @HostBinding('@routeAnimation')
 anyProperty = 'anything';

 constructor() { }

 ngOnInit() {
 }

}
```

This code applies the settings from the animation file to the class. It doesn't make a difference what property @HostBinding is applied to. Here its applied to one named 'anyProperty'.

__5. Save and close the file.

__6. Open the application in your browser. You may need to refresh again to see the animation.

```
http://localhost:4200
```

__7. Once the Home component is in view try clicking on one of the other buttons. You should notice the Home component slide out to the right when the next component is shown.

__8. Click on the 'Home' button and you should see the Home component slide in from the left. Notice though that the animation does not yet apply to the other two components, "About" and "Admin".

Part 6 - Add Animation support to the About and Admin components.

__1. Follow the instructions in Part 5 again, only this time apply them to the **About** component instead of the Home component.

```
\src\app\about\about.component.ts
```

__2. Do the same for the **Admin** Component.

```
\src\app\admin\admin.component.ts
```

Now that animation has been set up for all the components open the application and try using the buttons to navigate between them. You should see each component slide in from the left when they are invoked and out to the right when the next one is invoked.

__3. Stop the server and close the window.

__4. Close the browser.

__5. Close all open files.

Part 7 - Review

In this lab we have created animation settings and applied them to the components of our sample application. Animation used in this way can make applications more interesting and engaging for end users.