Angular Animations

Making Animations Work with Angular 4+

- First, **npm install --save @angular/animations**
- Within **AppModule**, import **BrowserAnimationsModule**

Animations, Triggers, and States

- If a component uses **animations**, it must include an **animations** array inside of its **@Component decorator**
- Every **animation** recognized by the **template** must be **included** in the **array**
- Every animation has a **trigger** function
    - **Trigger parameters**
        - A **name** to be recognized by the **template**
        - An **array** of **state** functions
            - **State parameters**
                - First: A **name**
                - Second: A **style** method **passing** some **CSS styling**
    - Added to the **template** as an element **attribute** via **property binding**
        - **[@divState]="<binding-condition>"**
- **Animations** are **movements** between **two states**

Switching Between States

- We should maintain a **state** field inside of our **component** that manages the **current animation state**
    - This is the same field recognized before in the **template** as **binding-condition**
- To **change** the **state**, we merely change that **field's value** to another **valid state**
    - **transition('normal <=> highlighted', animate(300)**

Transitions

- To **swap** between **states**, we add the **transition** function at the **same depth** of the **states**
    - **Transition parameters**
        - First: **'<starting-state => ending-state'**
        - Second: The **animate** function, accepting **animation conditions** as **arguments**

Advanced Transitions

- If we want **one transition** that goes in **both directions**, we use a **doubly-ended arrow** within the first argument
    - **transition('normal <=> highlighted', animate(300))**
- If we want a transition between one state and **any** state, we represent the **any** state with a **wildcard (*)**

Transition Phases

- We can define the **styles** an **animation** should take by **passing** them into the **animate** function
- Like with the **states**, pass the **style** function as a **second parameter**
    - It accepts an **object** of **CSS styles**
    - This can be jarring
- To make a more **fluid** animation, pass an array of **style** and **animate** methods as a **second parameter**

The "void" State

- **void** is a reserved **state** for cases where an **element** is in a **state** that **wasn't provided** by the **DOM**
- Since **void** is an **absence** of **state**, we must pass an **array** of **style** and **animation** functions

Using Keyframes for Animations

- **Keyframes** allow us to define states at specific times
    - Example → I want this state at XXX milliseconds, and this state XXX milliseconds later

- To do this, the **second parameter** within the **transition function** should be an **array** of **animate** and **keyframes** function calls
    - **keyframes** contains an **array** of **style** functions
    - All **styles/steps** take an **equal** amount of **time**
        - The **timing** can be **overridden** with the **offset** field

Grouping Transitions

- We can **group** transitions so they occur **simultaneously** (even with **offsets**)
- To do this, we use the **group** function with an **array** of **animate** function calls as an **argument**

Using Animation Callbacks

- We can **trigger** other **code** to **execute** upon an **animation's completion** using **callbacks**
- In the **template**, we use **event binding** that depend on our **states**
    - **(@state.start)="animationStarted($event)"** → Executes the function at the **beginning** of the **animation**
    - **(@state.done)="animationEnded($event)"** → Executes the function at the **end** of the **animation**