**ANIMATIONS**:

Animation provides the illusion of motion: HTML elements change styling over time. Well-designed animations can make your application more fun and easier to use, but they aren't just cosmetic. Animations can improve your app and user experience in a number of ways:

* Without animations, web page transitions can seem abrupt and jarring.
* Motion greatly enhances the user experience, so animations give users a chance to detect the application's response to their actions.
* Good animations intuitively call the user's attention to where it is needed.

Typically, animations involve multiple style transformations over time. An HTML element can move, change color, grow or shrink, fade, or slide off the page. These changes can occur simultaneously or sequentially. You can control the timing of each transformation

Angular's animation system is built on CSS functionality, which means you can animate any property that the browser considers animatable. This includes positions, sizes, transforms, colors, borders, and more. The W3C maintains a list of animatable properties on its CSS Transitions page.

* Import [BrowserAnimationsModule](https://angular.io/api/platform-browser/animations/BrowserAnimationsModule), which introduces the animation capabilities into your Angular root application module.
* In the component file, add a metadata property called animations: within the @component () decorator. You put the trigger that defines an animation within the animations metadata property.

**Animations transitions and triggers:**

Trigger defines the start of the animation. The first param to it is the name of the animation to be given to the html tag to which the animation needs to be applied. The second param are the functions we have imported - state, transition, etc.

This guide goes into greater depth on special transition states such as \* (wildcard) and void, and show how these special states are used for elements entering and leaving a view. This chapter also explores multiple animation triggers, animation callbacks, and sequence-based animation using keyframes.

### **Wildcard state:** An asterisk \* or wildcard matches any animation state. This is useful for defining transitions that apply regardless of the HTML element's start or end state.

For example, a transition of open => \* applies when the element's state changes from open to anything else.

In the component file, set the trigger that defines the animations as the value of the animations: property in the @component () decorator.

In the HTML template file, use the trigger name to attach the defined animations to the HTML element to be animated.

## **Animate multiple elements using query() and stagger() functions:**

The [query](https://angular.io/api/animations/query)() function allows you to find inner elements within the element that is being animated. This function targets specific HTML elements within a parent component and applies animations to each element individually. Angular intelligently handles setup, teardown, and cleanup as it coordinates the elements across the page.

The [stagger](https://angular.io/api/animations/stagger)() function allows you to define a timing gap between each queried item that is animated and thus animates elements with a delay between them.

The following example demonstrates how to use [query](https://angular.io/api/animations/query)() and [stagger](https://angular.io/api/animations/stagger)() functions on the entry of an animated element.

* Use [query](https://angular.io/api/animations/query)() to look for an element entering the page that meets certain criteria.
* For each of these elements, use [style](https://angular.io/api/animations/style)() to set the same initial style for the element. Make it invisible and use transform to move it out of position so that it can slide into place.
* Use [stagger](https://angular.io/api/animations/stagger)() to delay each animation by 30 milliseconds.
* Animate each element on screen for 0.5 seconds using a custom-defined easing curve, simultaneously fading it in and un-transforming it.

## **Creating reusable animations**

To create a reusable animation, use the [animation()](https://angular.io/api/animations/animation) method to define an animation in a separate .ts file and declare this animation definition as a const export variable. You can then import and reuse this animation in any of your app components using the [useAnimation()](https://angular.io/api/animations/useAnimation) API.

# Route transition animations:

The Angular router comes with high-level animation functions that let you animate the transitions between views when a route changes. To produce an animation sequence when switching between routes, you need to define nested animation sequences. Start with the top-level component that hosts the view, and nest additional animations in the components that host the embedded views.

**MATERIALS**:

Materials offer a lot of built-in modules for your project. Features such as autocomplete, datepicker, slider, menus, grids, and toolbar are available for use with materials in Angular 7.

To use materials, we need to import the package. Angular 2 also has all the above features but they are available as part of the **@angular/core module**. From Angular 4, Materials module has been made available with a separate module @angular/materials. This helps the user to import only the required materials in their project.

To start using materials, you need to install two packages: **materials and cdk**. Material components depend on the animation module for advanced features. Hence you need the animation package for the same, **@angular/animations**.

**Menu:**

To add menu, **<mat-menu></mat-menu>** is used. The **file** and **Save As** items are added to the button under mat-menu. There is a main button added **Menu**. The reference of the same is given the **<mat-menu>** by using **[matMenuTriggerFor]="menu"** and using the menu with **# in<mat-menu>**.

## **SideNav:**

To add sidenav, we need **<mat-sidenav-container></mat-sidenav-container>**. **<mat-sidenav></mat-sidenav>** is added as a child to the container. There is another div added, which triggers the sidenav by using **(click)="sidenav.open()"**.