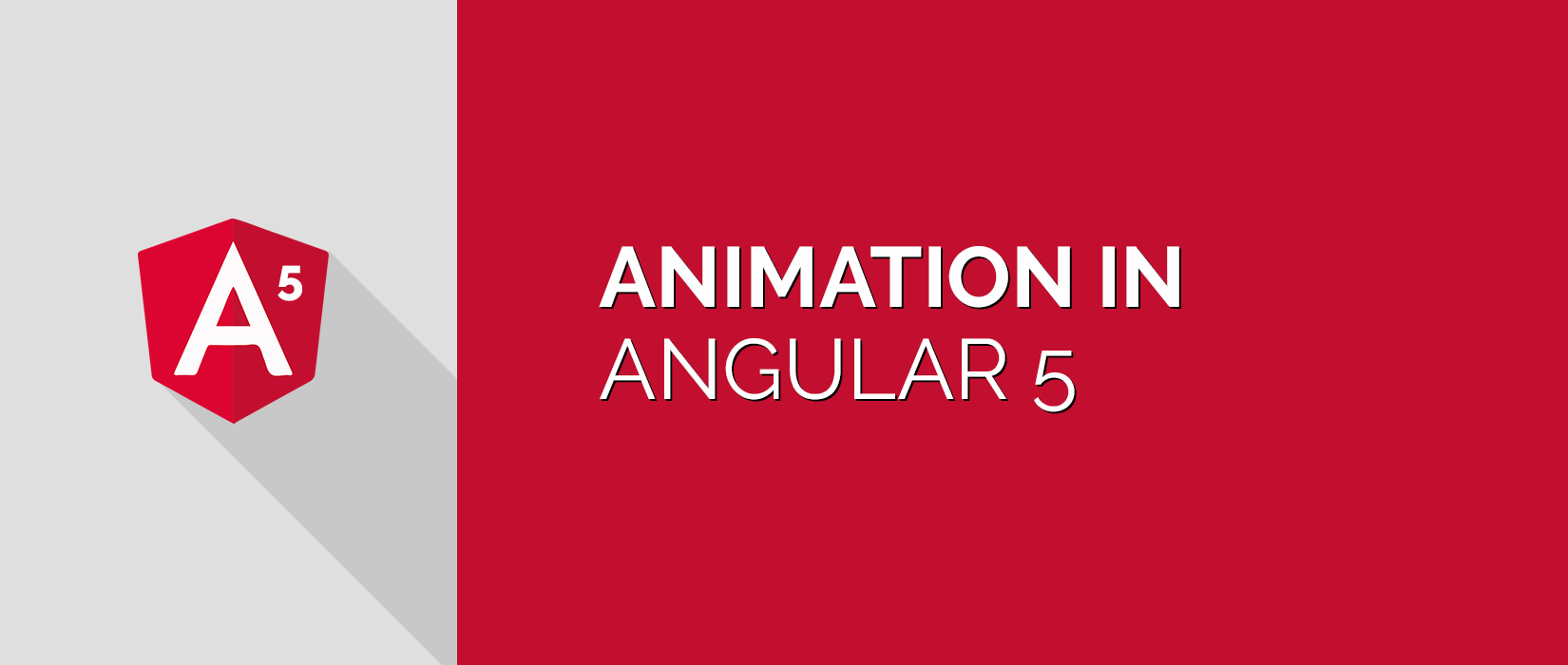


**Angular 5 Animation Tutorial**

BY GARY SIMON - NOV 03, 2017



The following tutorial is a part of our 100% free course [**Learn Angular 5 from Scratch - Angular 5 Tutorial**](https://coursetro.com/courses/19/Learn-Angular-5-from-Scratch---Angular-5-Tutorial)

In the [**previous lesson**](https://coursetro.com/posts/code/108/Angular-5-Interpolation,-Property-Binding-&-Event-Binding-Tutorial), we have much of our sample app working, which allows you to add a list of "life goals".

It would be awesome to integrate animation in such a way that once a new goal is added, it would animate in. We can also do other stuff, like animate things once they leave the DOM.

So, let's take a look at how we can integrate animation in an Angular 5 project.

**[[](https://coursetro.com/courses/19/Learn-Angular-5-from-Scratch---Angular-5-Tutorial?utm_source=in_article&utm_campaign=article&utm_medium=Learn+Angular+5+from+Scra)](https://coursetro.com/courses/19/Learn-Angular-5-from-Scratch---Angular-5-Tutorial?utm_source=in_article&utm_campaign=article&utm_medium=Learn+Angular+5+from+Scra)**

**[Learn Angular 5 from Scratch - Angular 5 Tutorial](https://coursetro.com/courses/19/Learn-Angular-5-from-Scratch---Angular-5-Tutorial?utm_source=in_article&utm_campaign=article&utm_medium=Learn+Angular+5+from+Scra)**

**[[](https://coursetro.com/courses/18/Build-a-Beautiful-CryptoCurrency-App-using-Ionic-3?utm_source=in_article&utm_campaign=article&utm_medium=Build+a+Beautiful+CryptoC)](https://coursetro.com/courses/18/Build-a-Beautiful-CryptoCurrency-App-using-Ionic-3?utm_source=in_article&utm_campaign=article&utm_medium=Build+a+Beautiful+CryptoC)**

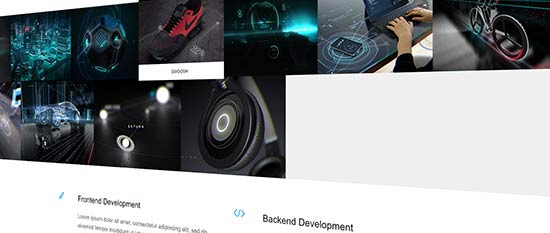
**[Build a Beautiful CryptoCurrency App using Ionic 3](https://coursetro.com/courses/18/Build-a-Beautiful-CryptoCurrency-App-using-Ionic-3?utm_source=in_article&utm_campaign=article&utm_medium=Build+a+Beautiful+CryptoC)**

**[[](https://coursetro.com/courses/13/Create-a-MEAN-App-Called-CodePost---Full-Stack?utm_source=in_article&utm_campaign=article&utm_medium=Create+a+MEAN+App+Called+)](https://coursetro.com/courses/13/Create-a-MEAN-App-Called-CodePost---Full-Stack?utm_source=in_article&utm_campaign=article&utm_medium=Create+a+MEAN+App+Called+)**

**[Create a MEAN App Called CodePost - Full Stack](https://coursetro.com/courses/13/Create-a-MEAN-App-Called-CodePost---Full-Stack?utm_source=in_article&utm_campaign=article&utm_medium=Create+a+MEAN+App+Called+)**

**[[](https://coursetro.com/courses/12/Learn-Angular-4-from-Scratch?utm_source=in_article&utm_campaign=article&utm_medium=Learn+Angular+4+from+Scra)](https://coursetro.com/courses/12/Learn-Angular-4-from-Scratch?utm_source=in_article&utm_campaign=article&utm_medium=Learn+Angular+4+from+Scra)**

**[Learn Angular 4 from Scratch](https://coursetro.com/courses/12/Learn-Angular-4-from-Scratch?utm_source=in_article&utm_campaign=article&utm_medium=Learn+Angular+4+from+Scra)**

**[[](https://coursetro.com/courses/10/Create-a-Personal-Portfolio-using-Angular-2-&-Behance?utm_source=in_article&utm_campaign=article&utm_medium=Create+a+Personal+Portfol)](https://coursetro.com/courses/10/Create-a-Personal-Portfolio-using-Angular-2-&-Behance?utm_source=in_article&utm_campaign=article&utm_medium=Create+a+Personal+Portfol)**

**[Create a Personal Portfolio using Angular 2 & Behance](https://coursetro.com/courses/10/Create-a-Personal-Portfolio-using-Angular-2-&-Behance?utm_source=in_article&utm_campaign=article&utm_medium=Create+a+Personal+Portfol)**

**If you prefer watching a video instead..**

Be sure to **[Subscribe to the Official Coursetro Youtube Channel](http://youtube.com/user/designcourse)** for more videos.

**Installing the Animation Library**

Until recently, the Angular Animation library was installed by default. But now, we have to add it manually.

Visit the console in the project folder and type:

$ npm install @angular/animations@latest --save

Next, in */src/app/app.module.ts* we need to import it:

// Other imports removed for brevity

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

@NgModule({

...

imports: [

// other modules removed for brevity

BrowserAnimationsModule

],

})

**Importing Animations**

We want to add animations to our home component. So, in the */src/app/home/home.component.ts* file, import at the top:

import { trigger,style,transition,animate,keyframes,query,stagger } from '@angular/animations';

**Defining Animations**

Animations are defined within the @Component decorator like so:

@Component({

selector: 'app-home',

templateUrl: './home.component.html',

styleUrls: ['./home.component.scss'],

animations: [

// Animations here...

]

})

The first animation-specific function that we use is **trigger()**. This allows us to define the name of the animation in the first parameter, and then an array of other animation-specific functions in the next.

animations: [

trigger('goals', [

])

]

We've made a new animation called **goals** here. Next, we use **transition()** to define a transition between two different **states**. States can be defined explicitely through the **state** function, or we can use a wildcard \* as a state.

There are a variety of transition states, but the one we're going to use is \* => \*:

animations: [

trigger('goals', [

transition('\* => \*', [

])

])

]

This means, for the animation trigger *goals*, if there is any transition taking place, execute the following..

Next, we're going to use the **query** function to target DOM elements that are only entering the DOM:

animations: [

trigger('goals', [

transition('\* => \*', [

query(':enter', style({ opacity: 0 }), {optional: true}),

query(':enter', stagger('300ms', [

animate('.6s ease-in', keyframes([

style({opacity: 0, transform: 'translateY(-75%)', offset: 0}),

style({opacity: .5, transform: 'translateY(35px)', offset: 0.3}),

style({opacity: 1, transform: 'translateY(0)', offset: 1.0}),

]))]), {optional: true})

])

])

]

Wow, that's a lot!  Well, first, we're setting any DOM element that is bound to the **goals** animation, to an opacity of 0 when it enters.

Next, we're creating a **stagger** animation with a delay of 300ms between each element.

After that, we're animating the elements with a *.6s ease-in* animation, and using **keyframes** to allow us to create a multi-step animation for each item.

Is all of this necessary? No, you could get rid of the stagger and the keyframes, but this more complex approach will show you the full power of Angular 5 Animations.

**Applying the Animation in the Template**

Before we add to the template, let's give our default **goals[]** array some initial values:

goals = ['My first life goal', 'I want to climb a mountain', 'Go ice skiing'];

Now that we have our animations defined in the component, let's make the following adjustments in the *app.component.html* template:

<!-- From: -->

<div class="container color-light">

<!-- To: -->

<div class="container color-light" [@goals]="goals.length">

Great! Save it and reload the browser.

You should see each of the 3 goal items animate in a unique stagger.  Try adding a goal yourself with the form, and you will see it too animates in as expected.

How about animating a list item when it's removed from the DOM?

In the template, add a click event here:

<!-- From: -->

<p class="life-container" \*ngFor="let goal of goals" (click)="removeItem(i)">

<!-- To: -->

<p class="life-container" \*ngFor="let goal of goals; let i = index" (click)="removeItem(i)">

In the component class, add the *removeItem()* method:

removeItem(i) {

this.goals.splice(i, 1);

}

Then, in the animations above, add the following:

animations: [

trigger('goals', [

transition('\* => \*', [

query(':enter', style({ opacity: 0 }), {optional: true}),

query(':enter', stagger('300ms', [

animate('.6s ease-in', keyframes([

style({opacity: 0, transform: 'translateY(-75%)', offset: 0}),

style({opacity: .5, transform: 'translateY(35px)', offset: 0.3}),

style({opacity: 1, transform: 'translateY(0)', offset: 1.0}),

]))]), {optional: true})

,

query(':leave', stagger('300ms', [

animate('.6s ease-out', keyframes([

style({opacity: 1, transform: 'translateY(0)', offset: 0}),

style({opacity: .5, transform: 'translateY(35px)', offset: 0.3}),

style({opacity: 0, transform: 'translateY(-75%)', offset: 1.0}),

]))]), {optional: true})

])

])

]

Notice we've added a query for **:leave**? It's as simple as that, and then reversing some of the style values to produce the opposite effect of opacity and Y movements.

Save it, and try clicking on any of the goal items in the list. They will animate out!

**Going Forward**

There is, of course quite a bit more to Angular animations, but this should be a good primer on how to use them.

In the next lesson, we're going to take a look at the Angular 5 Router.