Angular

Styles & Animations

Getting Started

What to expect from this course

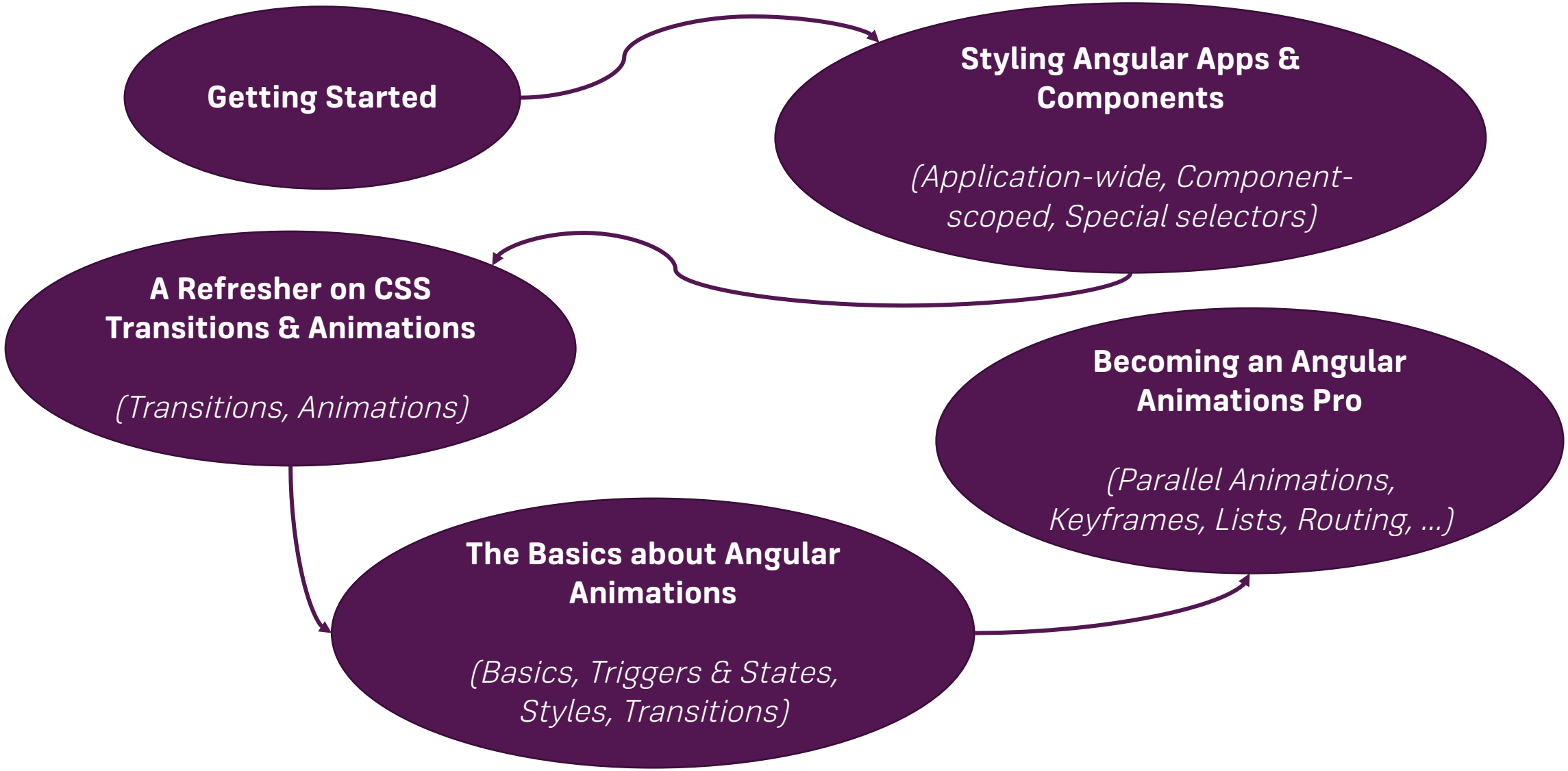
This Course is about ...

- Styling Angular Apps & Components
- Tools Angular gives you to Style your Apps, Components and HTML Elements
- How to use CSS Transitions and Animations in Angular Apps
- Understanding the Angular Animation Package In-Depth
- Practicing all these Things

This Course is not about ...

- Learning CSS
- Learning CSS Transitions & Animations
- Learning or Using SASS, SCSS, LESS...
- Learning Angular

Course Structure



How to get the Most out of the Course

Watch the Videos

Do the Course Project

Ask in Q&A, but ...

...also answer in Q&A!

Styling Angular Apps

How to make your apps look beautiful

Using "Normal" CSS

Application-wide Styles

Import in index.html
Import (via Webpack) in other Files

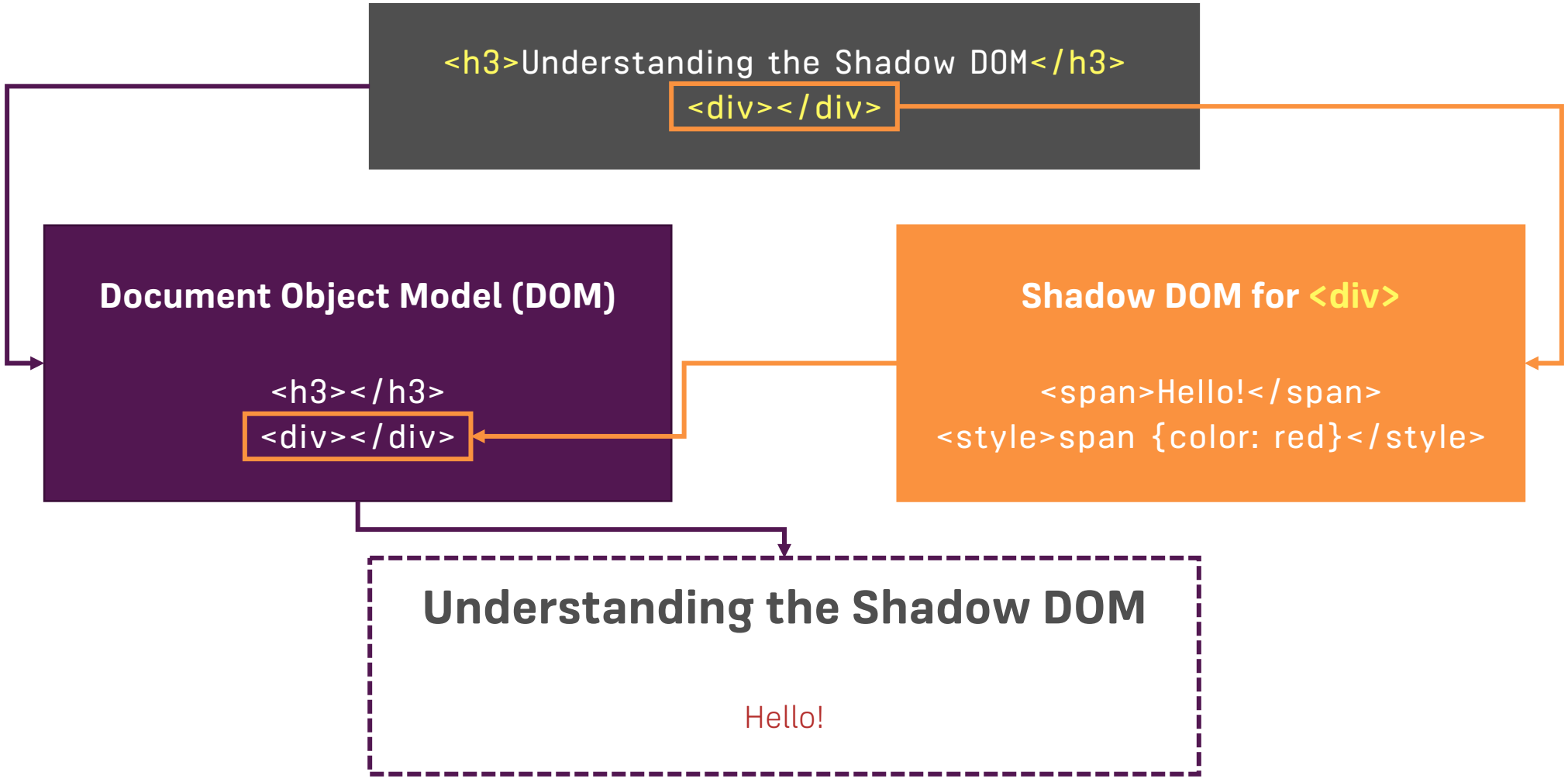
Component-scoped Styles

Use styles or styleUrls property in @Component Decorator
Add <style> or <link> Tags to your Component Templates

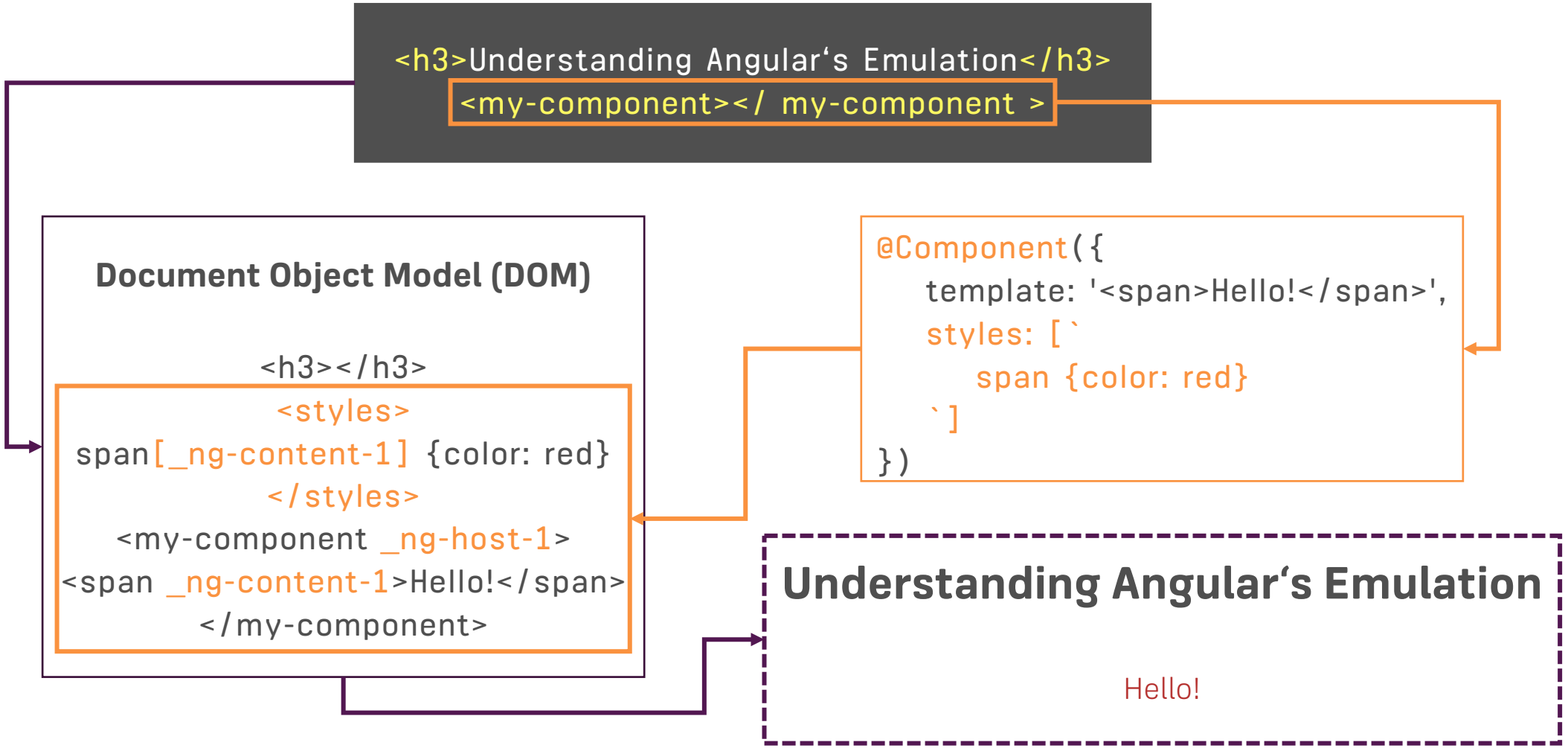
Understanding View Encapsulation

Angular emulates the Shadow DOM and therefore encapsulates Styles defined in styles or styleUrls on component level.

Understanding the Shadow DOM



Understanding Angular's Emulation



Understanding Angular's Emulation

```
@Component: {
    …
    encapsulation: ViewEncapsulation.Emulated
}
```

◀ Default

```
ViewEncapsulation.Native
```

◀ Uses native Shadow DOM

```
ViewEncapsulation.None
```

◀ Styles are shared across Components

Special Selectors

`:host`	Targets the element which hosts the Component
`:host-context`	Style elements inside a component, depending on some condition set outside of it
`/deep/`	Apply styles not just to the elements in the template but also to child components

ngClass

Conditionally apply CSS Classes to Elements

```
[ngClass]="{class1: condition}"

[ngClass]="'class1 class2'"

[ngClass]="[class1 class2]"

[ngClass]="{'class1 class2': condition}"
```

Or use any expression (e.g. function call) which returns on of the valid values!

ngStyle

Dynamically apply CSS Styles to Elements

```
[ngStyle]="{'background-color': 'red'}"

[ngStyle]="{'width.px': 200}"
```

Or use any expression (e.g. function call) which returns on of the valid values!

Using CSS Animations

How and When to use them

CSS Transition Property

```
div {
    transition: width 0.3s ease-out;
}
```

Whenever the width of this changes, the change is going to get animated over 0.3 seconds (with an ease-out timing function)

CSS Animations

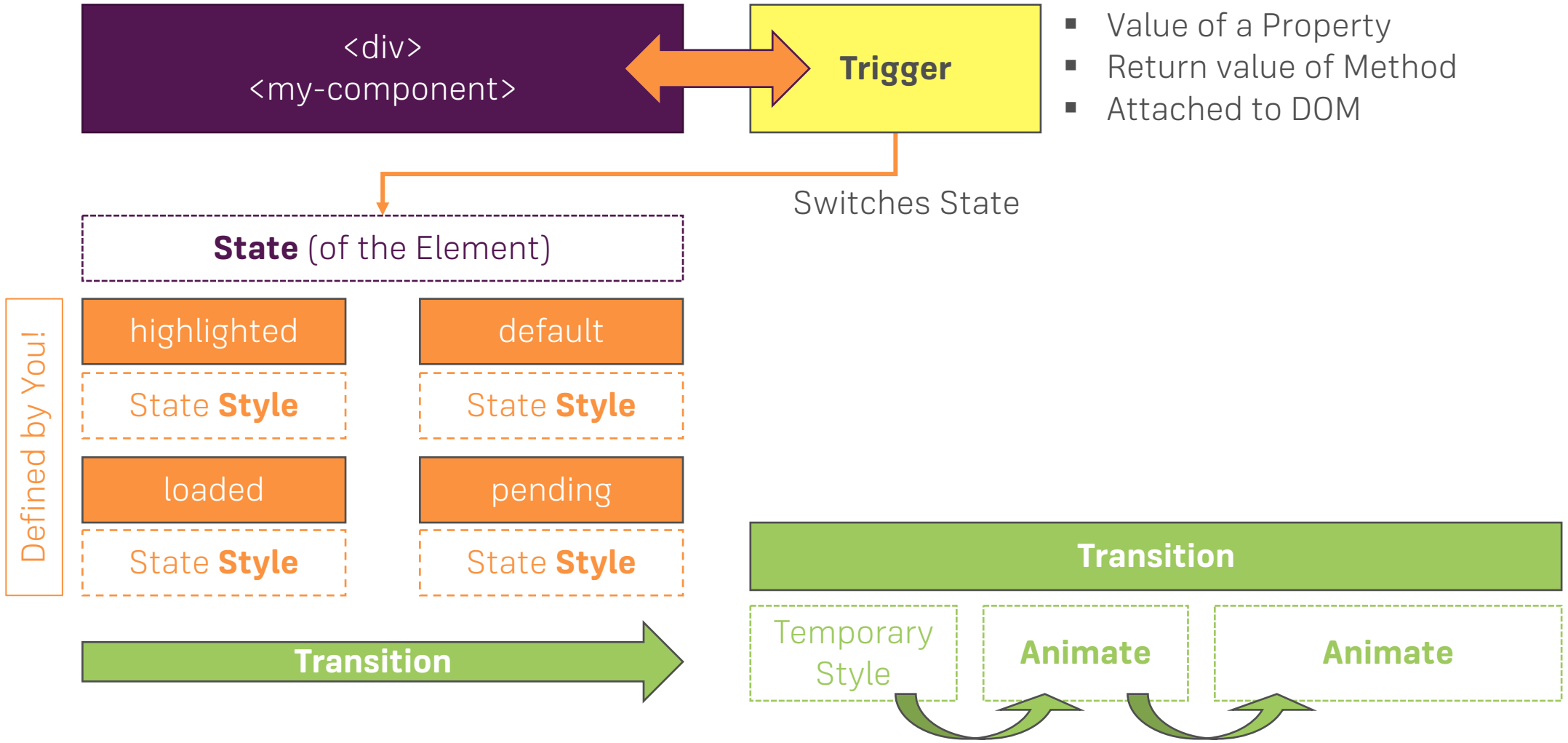
```
div {
    animation: myAnimation 0.3s ease-out forwards;
}

keyframes myAnimation {
    0% {
        opacity: 0
    }
    100% {
        opacity: 1
    }
}
```

Animates the opacity of the div over a duration of 0.3 seconds (with an ease-out timing function) from 0 (fully opaque) to 1 (fully visible) and keeps the final animation status as the element style.

The Basics About the Angular Animations Library

How to use the Angular Animations Library

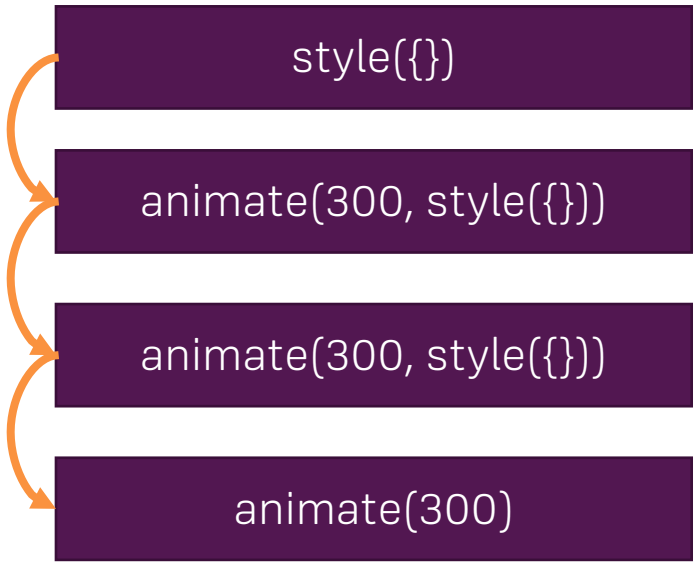
How Angular Animations Work



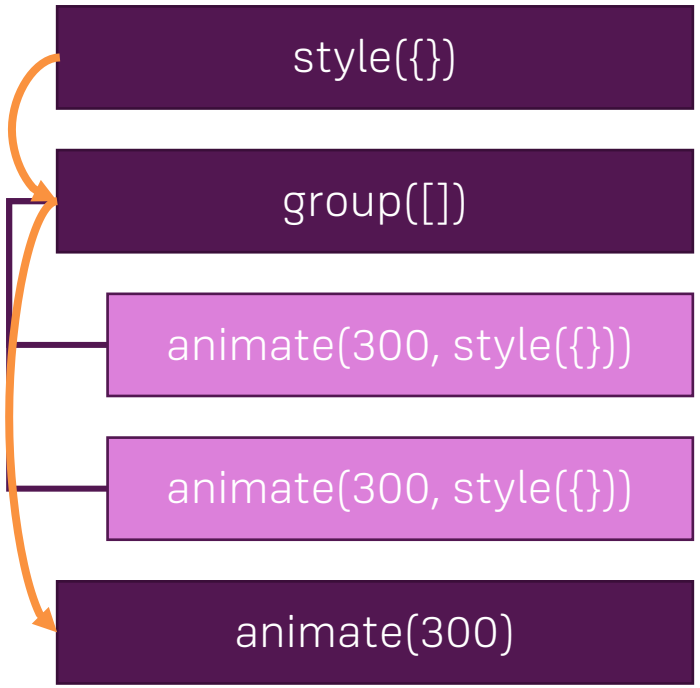
Becoming an Angular Animations Pro

Diving deeper into the Angular Animations Library

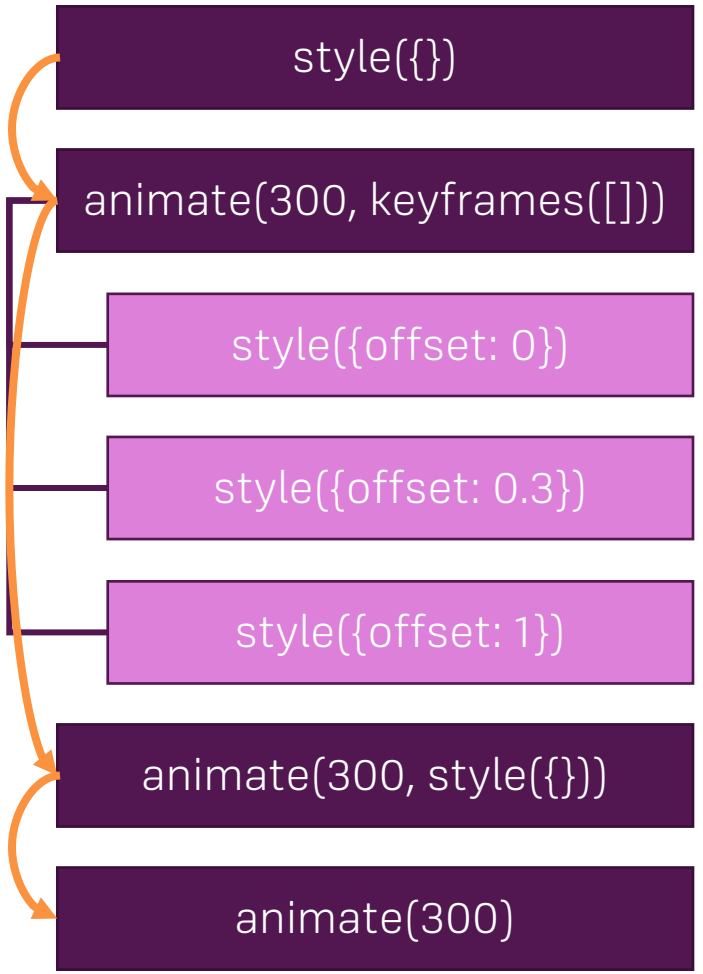
Multi-Step vs group() vs keyframes()



3 Steps (900ms)



2 Steps (600ms)



3 Steps (900ms)

Animating Route Transitions

Show the User what‘s happening!

Animating Routing