**Angular 4 CSS**

When you use the Angular CLI to generate an Angular project, by default it will setup the project to work with standard CSS.

However, if you wish to use Sass or SCSS, you add the --style prefix as such:

// Starting a project with standard CSS

> ng new project-name

// Starting a project with Sass

> ng new project-name --style=sass

// Starting a project with SCSS

> ng new project-name --style=scss

We already generated a project with standard CSS in this course, so we will continue on assuming you generated it without the **--style** prefix.

## Inline CSS vs. External CSS Stylesheets

When you use the Angular CLI to generate a project, by default it will setup your components to work with external stylesheets.

Whether you're using inline or external CSS, they're both defined within the component decorator.

An external stylesheet is defined by the **styleUrls**property:

@Component({

// Other properties removed

styleUrls: ['./app.component.css']

})

This way, all of the CSS that's specific to this component's template, will be placed in a separate CSS file. This is useful when you're dealing with a lot of CSS and do not want that CSS cluttering your component code.

Should you prefer to write inline CSS within the component decorator, you can change the **styleUrls** property to **styles**.

@Component({

// Other properties removed

styles: [`

h1 {

text-decoration:underline;

}

`]

})

This way, we can use backticks to allow us to define multi-line CSS, and our CSS declarations are written inline. This is useful when you have very few CSS declarations, or just prefer to write inline CSS within your component.

## Changing CSS through Class Binding

Sometimes you need to change CSS on the fly. Angular allows you to add or remove CSS classes based on component logic.

To change CSS classes, in your **app.component.html** template:

<h1 [class]="titleClass">

{{title}}

</h1>

Then, in the component class we define a **titleClass** property:

export class AppComponent {

title = 'Hello!';

titleClass = 'red-title';

}

As you can see, we've bound **titleClass** to a string called **red-title**, which is the name of a CSS class that we have to define. So, in the **styles** property, we can define that class:

styles: [`

h1 {

text-decoration:underline;

}

.red-title {

color:red;

}

`]

Now, if you run **ng serve** at the console and view the project at **http://localhost:4200**, you will see that the **Hello!** text is red.

Class binding is useful in a real-world context because it allows you to control which CSS class is attached to the property that **[class]** is bound to.

You can also designate the CSS class right in the class binding within the template:

<h1 [class.red-title]="titleClass">

{{title}}

</h1>

Then, **titleClass** in your component code can either be bound to some value, or **false**, which will result in the class being removed from the element. Try changing **titleClass** to false and the title will no longer remain red.

## Changing CSS through Style Binding

Changing the style attribute is just another form of property binding. So, using our example above, change the template to:

<h1 [style.color]="titleStyle">

{{title}}

</h1>

Then, change our **titleClass** property in the component class to:

titleStyle = 'red';

You can also control the value of the specified CSS property, within the template expression itself:

<h1 [style.color]="titleStyle ? 'green' : 'pink'">

{{title}}

</h1>

This way, if **titleStyle** is defined, it will appear green. Otherwise, if it does not exist, or is bound to **false**, it will appear pink.

## Changing Multiple CSS Styles & Classes

You can bind the **ngStyle** and **ngClass** directives to a property. This property can control multiple styles or classes.

Change the template HTML to:

<h1 [ngClass]="titleClasses">

{{title}}

</h1>

Then, change the component class to:

titleClasses = {

'red-title': true,

'large-title': true

}

In the styles property of the component decorator, add the **large-title** class:

.large-title {

font-size:4em;

}

Now, both of the classes that are defined as being **true** in the **titleClasses** object will be attached to the h1 element.

It's the same approach for changing multiple styles through the **ngStyle** directive:

<h1 [ngStyle]="titleStyles">

{{title}}

</h1>

And in the component class:

titleStyles = {

'color': 'red',

'font-size' : '4em'

}