MODULE 01 - 011: SCSS Nested styles

One of the most powerful features of SCSS is **style nesting**, making stylesheets cleaner and more intuitive.

SCSS provides an intuitive nesting feature, allowing you to define styles within the context of their parent.

Why Use Nesting?

In traditional CSS, defining styles for deeply nested DOM elements can become so complicated:

```
.my-class > .etc-class {
  color: DarkRed;
}
```

But what if you have complex nested elements, such as:

```
1. A container for the entire page
2. A specific container for code snippets
```

Then, the common problems with traditional CSS:
- You'd have to create and manage separate classes. - Any updates require careful synchronization across stylesheets.

And, here comes SCSS.

SCSS allows to nest .classes inside another .classes, overriding its global scope

SCSS nesting is an excellent way to:

- Organize styles hierarchically.
- Ensure cleaner, more readable code.
- Avoid repetitive declarations for complex structures.

```
$off-white: #f6f6f6;
$featured-color: DarkRed;

body {
  background-color: $off-white;
  height: 100vh;
  height: 100vw;
}

.container {
  font-family: Verdana;
  font-size: 0.8rem;
}

.page-wrapper {
  padding: 21px;
  $featured-color: LightCoral;

  .featured {
    color: $featured-color;
    .subheading {
      color: Red;
    }
  }
```

```
  .page-content {
    background-color: $featured-color;
    padding: 42px;
    color: $off-white;

    .container {
      font-family: courier;
    }
  }
}
```

Observations

- The `.container` class inside `.page-content` **overrides the outer container's styles**, making it more specific.

Video lesson Speech

[ENG] # Guide to SCSS Nesting One of the most popular features of SCSS is the ability to nest style definitions, and that's what we're going to walk through in this guide. ***

In traditional CSS it's possible to say that you want to define a set of styles for one or more classes, such as with code like this:

```
.my-class > .etc-class {
  color: DarkRed;
}
```

However, it gets trickier when you want to implement styles for DOM elements that are nested within other elements. For example, if you want to have two container classes on a page:

```
1.  For the entire page
2.  For spots on the page that have code snippets
```

You might think that you'd need to rename the container classes to have two separate classes.
However, that would mean that each time you alter the styles in one container you'll have to remember to go change each of the other similar classes.
There are a few ways to share the style definitions in pure CSS (remember that SCSS is simply a translator into CSS), however, it can be a bit complex.

Thankfully SCSS offers an intuitive way to nest styles.

In the example below, we have two container classes. Notice how we have one 'master' container class definition that has a font family and font size. And then if you traverse down to the inside of the page-content class I've nested another call to the container class. However, this nested container class is more specific, so the styles that are defined in it will only apply to the DOM elements that are stored inside of the container class nested inside of the page-content div.

```
$off-white: #f6f6f6;
$featured-color: DarkRed;

body {
  background-color: $off-white;
  height: 100vh;
  height: 100vw;
}

.container {
  font-family: Verdana;
  font-size: 0.8rem;
}

.page-wrapper {
```

```
  padding: 21px;
  $featured-color: LightCoral;

  .featured {
    color: $featured-color;
    .subheading {
      color: Red;
    }
  }

  .page-content {

    background-color: $featured-color;
    padding: 42px;
    color: $off-white;
    .container {
      font-family: courier;
    }
  }
}
```

[SPA]

Los estilos anidados en SCSS son una herramienta mejorada del CSS tradicional, evitando el caos de los selectores largos y complicados de ésta.

¿Cómo?

SCSS ofrece una solución mucho más sencilla:

Anidar estilos dentro de estilos