Hello and welcome to **A to Z Sass** and to a brand new video series. Over the next 26 episodes, we're going to be diving into the CSS pre-processor, **Sass**, and looking at a different topic, or a different concept, or a different feature or tool.

In this lesson let's talk about the Sass ampersand character `&`. This single character is a powerful part of the Sass language and has a number of different uses. In this episode, you'll also learn about nesting and the Sass parents selector, the reverse parent selector and the double ampersand selector.

Nesting is a popular feature in Sass, but it's something to do infrequently and something to do carefully to avoid unnecessarily high specificity in your selectors when they're compiled into CSS. Nesting allows us to write a shorter syntax, which is then compiled into normal CSS.

```
.site-nav {
    ul {
        li {
        }
    }
}
```

Here, the `ul` and `li` are indented inside the braces of the `.site-nav` class. And when compiled, this would produce the following CSS:

```
.site-nav ul {
}

.site-nav  ul li {
}
```

There is also a way to reference the parent selector within Sass nesting, and we do that with the ampersand character. The ampersand is a bit like a special variable that always holds the value of the current selector's parent. While nesting is a useful way to avoid typing out repetitive selectors, it should be used with caution. Try to limit yourself to only nesting with pseudo selectors where possible. But at most, try to not go more than three levels deep to avoid these overly specific selectors.

```
.site-nav {
    ul {
        li {
        }
    }

    a {
        &:hover {
            color: red;
```

```
        }
    }
}
```

This will make all hovered links inside the `.site-nav` red.

The ampersand character has another use in something that I call the **reverse parent selector**. Imagine a project which has got multiple color schemes that are all switched via a class on the `body` tag. Now, imagine that each of these color schemes or themes have different colored links and different colored buttons.

So we could setup some default styles as follows. And then, to change these styles for each theme, we can use the Sass ampersand as follows. Here, the ampersand appears at the end of the selector, but it still serves the same purpose, which is to output whatever the value of the parent selector is in its place.

So with this reverse parent selector in place, this is what our compiled CSS would look like. This use of the ampersand is really handy for situations where a parent class is used to overwrite styles. And using it in this way helps us to keep all the related styles together, making the code more compact and easier to read without scrolling up and down the file.

One final use of the Sass ampersand is the **double ampersand selector**. Here's an example of what that might look like in practice. This uses an adjacent sibling selector shown by the plus character and would compile as follows. So this selector will match any button class that immediately follows another button class.

But how is that useful? Well, if we think of a series of buttons that layout horizontally, we may want to space them out a little bit with some margin. But we'd always end up having to remove some margin from the first one. Or remove some margin from the last one using first child or last child.

Instead, we can use this double ampersand to only add margin left to buttons that are adjacent to other buttons. It may look a little odd to begin with, but the end result is much less code and removes the need for overriding stars that have already been declared. And this is the real power of Sass, which for us to write less and have it do more.