Control Directives

In this lesson, we'll be looking at SASS control directives, which allow us to introduce conditionals and looping into our stylesheets.

Definition

Control directives and **expressions** are used in SASS to include styles only under certain defined conditions.

As a feature, they're quite advanced and are mainly useful in mixins. Common directives include `@if`, `@else`, `@for` and `@while`.

@if and @else

The `@if` and `@else` directives are similar to `if` and `else` statements in JavaScript.

`@if` takes an expression and executes the styles contained within its block — if the evaluation **is not false** (or null).

`@else` will then be checked, if the previous `@if` evaluated to false.

For example:

Output

HTML

CSS (SCSS)

```
@mixin heading($size) {
  @if $size == large {
    font-size: 4rem;
  }
  @else if $size == medium{
    font-size: 3rem;
  }
  @else if $size == small {
    font-size: 2rem;
  }
  @else {
    font-size: 1rem;
  }
}

.h1 {
  @include heading(large);
}

.h6 {
  @include heading(small);
}
```

Here, we are using a **heading** mixin which accepts `$size` as an argument. We can have a different size for each of our headings depending on which value we pass to the mixin.

@for

You can use the `@for` directive to execute a group of statements a specified number of times. Effectively this is a loop.

It has two variations. The first uses the `through` keyword, it'll execute the statements from **start** to **end**, inclusive.

'through' example:

```
@for $i from 1 through 5 {
    .list-#{$i} {
       width: 2px * $i;
    }
}
```

This will produce the following CSS output:

```
.list-1 {
  margin-left: 2px;
}

.list-2 {
  margin-left: 4px;
}

.list-3 {
  margin-left: 6px;
}

.list-4 {
  margin-left: 8px;
}

.list-5 {
  margin-left: 10px;
}
```

If we replace the `through` keyword with `to`, it makes the loop exclusive. The difference being that it won’t execute when the variable is equal to **end**.

’to’ Example:

```
@for $i from 1 to 5 {
   .list-#{$i} {
      width: 2px * $i;
   }
}
```

Would yield the following CSS:

```
.list-1 {
  margin-left: 2px;
}

.list-2 {
  margin-left: 4px;
}

.list-3 {
  margin-left: 6px;
}


.list-4 {
  margin-left: 8px;
}
```

@while

We could instead implement the above code using the `@while` directive. As its name implies, it will continue to output CSS produced by the statements while the condition returns true.

The syntax is as follows:

```
$i: 1;
@while $i < 6 {
  .list-#{$i} {
     width: 2px * $i;
  }
  $i: $i + 1;
}
```

The output is identical, so you could opt for your personal preference based on the syntax.

Next, we'll look at SASS interpolation.