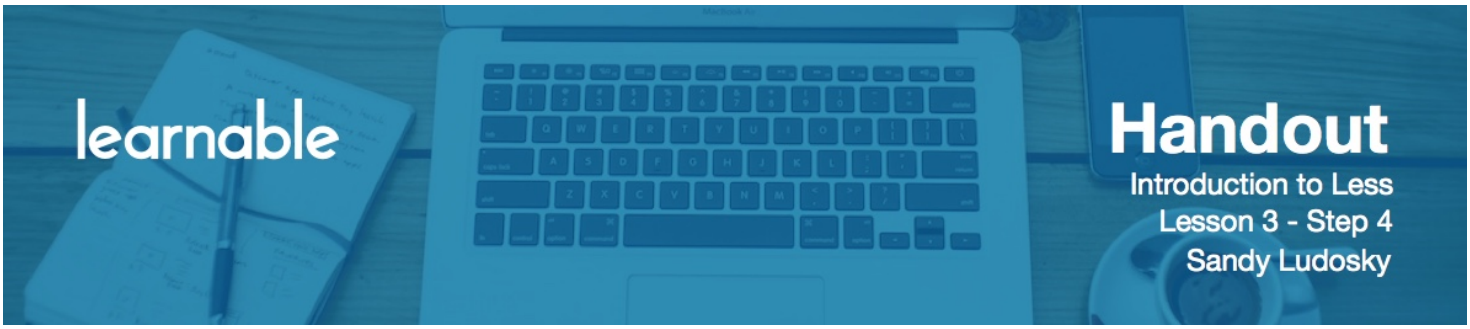

What are Mixins?

In this step we're going to start talking about the **mixins**. The mixins are like re-usable objects, that include a set of rules, that can be reused multiple times in your stylesheets. That allows to serve the same purpose of keeping the code DRY, clean and better organized.

Custom Mixins

Here is the example of a mixin:

```
.bordered {
  border-top: @border;
  border-bottom: @border;
}
```

This includes two CSS properties: `border-top` and `border-bottom`. The variable `@border` itself includes three different values:

```
@border: 4px solid @dark;
```

Now let's add top and bottom borders the `h1` by using the mixin:

```
h1 {
    color: @lightgrey;
    padding: @padding;
    margin: 0;
    .bordered;
}
```

Rremember to always finish a line with a semicolon.

Apply the same styling for the footer:

```
footer {
    .bordered;
}
```

But that's not enough. I'd like to have the exact same styling for the `footer` that we have for the `h1` apart from the padding:

```
footer {
    color: @lightgrey;
    margin: 0;
    .bordered;
}
```

However what we have done is we repeated ourselves. What we should do instead is to declare a mixin and then reuse it multiple times.

How mixins work?

Let's name it `commonRules`:

```
.commonRules {
    padding: @padding;
    margin: 0;
    color: @lightgrey;
    background:@darkgrey;
    text-align: right;
```

```
    .bordered;
}
```

Now you can reuse it multiple times in your Less code this mixin:

```
header {
    h1 {
        .commonRules;

        &:after {
            content: @stringVar;
        }
    }
}

footer {
  .commonRules;
  padding: 0 @padding;
  position: absolute;
  left: 0;
  right: 0;
  bottom: 0;
}
```

Now both the `header` and the `footer` will inherit from the CSS properties of the `commonRules` mixin. I had to apply additional padding for the `footer` to redefine padding provided by the mixin. This is the principle of the cascading stylesheet meaning that whichever comes last, will take precedence.

So that was one example of a mixin that you can declare once in your Less code and then reuse multiple times as an object.