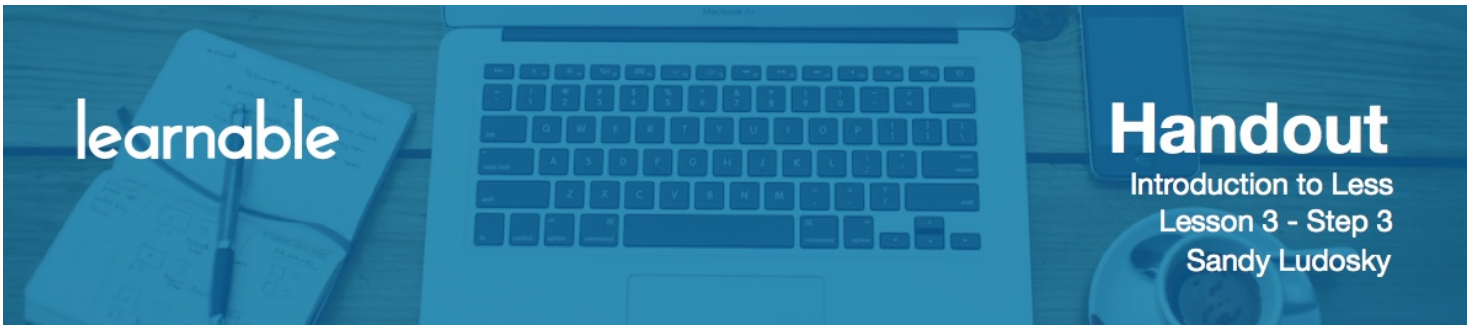

Introduction

In this step we will discuss **nesting**. Nesting allows you to nest selectors within each other in order to keep the code clean and well organized.

Nesting in Less

So as you remember the purpose of Less is to write less code and to keep the code DRY. But by looking at the code inside *style.css* looks like there is a lot of repetition and duplication of selectors. Luckily, with nesting, which is another great feature of Less, we can nest selectors within each other, following the same structure of the **DOM (document object model)**.

Start with the `header`:

```
header {
    background:@darkgrey;

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

Inside the `header` we're referencing `h1` by using nesting that allows us to keep the code clean and well organized. Check out the output in CSS: the Less compiler understood that the `h1` was part of the header. This way we didn't need to write the same selector two times.

Now I want to pan some content next to the `h1`:

```
header {
    background:@darkgrey;

    h1 {
        color: @lightgrey;
        padding: @padding;
        margin: 0;

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

For this I am using the combinator ampersand, followed by the pseudo class after. Using the ampersand combinator tells Less to concatenate one selector with the parent selector with no space in the middle. This is always useful when you want to use pseudo class like hover or focus. We're going to see more example of that later.

Now let's look at the navigation section:

```
nav {
  background: @mediumgrey;

  ul {
    margin: 0;

    li {
```

```
            display: inline-block;
            font-size: @fontSize;
            margin: @margin;

            a {
                text-decoration: none;
                color: @lightgrey;
            }
        }
    }
}
```

Once again we are following the structure of the DOM of the HTML page. So it allows to keep things simple, and also to make inheritance clearer.