# Using LESS

Using LESS is really easy. We only need this two lines inside the head tag of your HTML document. First, the .less file then the less.js library at the bottom (which you can download it from the official site).

```
<link rel="stylesheet/less" type="text/css" href="style.less">
<script src="less.js" type="text/javascript"></script>
```

However, this time we will show you how to do it the other way. We are going to use a LESS compiler, you can use WinLess for Windows or LESS.app if you are running on Mac OSX. Go to the site, download and install it in your machine.

Then, create a .less file in your working directory.

Open the compiler (I'm referring to the WinLESS or LESS.app), and import your working directory into it, the compiler will then find any .less file inside the directory.

LESS CSS Simple LESS

```
$ sudo npm install -g less
```

That's it. We are all set up now, and whenever we make a change and save the .less file, the compiler will automatically generate the regular CSS in .css format which the browser can read in the directory.

Now, we only need to link the CSS file to our HTML document, as follows;

```
<link rel="stylesheet/css" type="text/css" href="style.css">
```

## LESS Syntax

Unlike regular CSS as we know it, LESS works much more like a programming language. It's dynamic, so you'll expect to find some terminologies like Variables, Operation and Scope along the way.

## Variables

First of all, let's take a look at the Variables.

If you've been working quite long with CSS, you probably often written something like this, where we have repetitive values assigned in some declaration blocks in the entire stylesheet.

```
.class1 {
    background-color: #2d5e8b;
}
.class2 {
    background-color: #fff;
    color: #2d5e8b;
}
.class3 {
    border: 1px solid #2d5e8b;
}
```

This practice is actually fine, until we find ourselves having to sift through more than 1000 more similar snippets throughout the stylesheet. This could happen when we build a large scale website, and then things become so tedious when we begin to modify it.

If we are using a CSS pre-procesor like LESS, the instance above would not be a problem, as we can use Variables. The variables will allow us to store a constant value that later can be reused in the entire stylesheet.

```
@color-base: #2d5e8b;

.class1 {
    background-color: @color-base;
}
.class2 {
    background-color: #fff;
    color: @color-base;
}
.class3 {
    border: 1px solid @color-base;
}
```

In the example above, we store the color #2d5e8b in @color-base variable. Then, just in case you want to change the color, we only need to change the value in the variable.

You can also put other values in the variables like the examples below:

```
@font-family: Georgia
@dot-border: dotted
@transition: linear
@opacity: 0.5
```

## Mixins

In LESS, we can use Mixins to reuse whole declarations in a CSS rule set to another rule set. Here is an example;

```
.gradients {
    background: #eaeaea;
    background: linear-gradient(top, #eaeaea, #cccccc);
    background: -o-linear-gradient(top, #eaeaea, #cccccc);
    background: -ms-linear-gradient(top, #eaeaea, #cccccc);
    background: -moz-linear-gradient(top, #eaeaea, #cccccc);
    background: -webkit-linear-gradient(top, #eaeaea, #cccccc);
}
```

In the above snippet, we have preset a default gradients color inside .gradients class. Then, whenever we want to add the gradients we simply insert the .gradients this way:

```
div {
    .gradients;
    border: 1px solid #555;
    border-radius: 3px;
}
```

The .box will inherit all the declaration block inside the .gradients. So, the CSS rule above is equal to the following regular CSS;

```
div {
    background: #eaeaea;
    background: linear-gradient(top, #eaeaea, #cccccc);
    background: -o-linear-gradient(top, #eaeaea, #cccccc);
    background: -ms-linear-gradient(top, #eaeaea, #cccccc);
    background: -moz-linear-gradient(top, #eaeaea, #cccccc);
    background: -webkit-linear-gradient(top, #eaeaea, #cccccc);
    border: 1px solid #555;
    border-radius: 3px;
}
```

Furthermore, if you are using CSS3 a lot in your website, you can use LESS ELements to make your job much easier. LESS Elements is a collection of common CSS3 Mixins that we may use often in stylesheets, such as border-radius, gradients, drop-shadow and so on.

To use LESS Elements, simply add the @import rule in your LESS stylesheet, but don't forget to download it first and add it into your working directory.

```
@import "elements.less";
```

We can now reuse all the classes provided from the elements.less, for example, to add 3px border radius to a div, we can write

```
div {
    .rounded(3px);
}
```

For further usage, please refer to the official documentation.

## Nested Rules

When you write styles in plain CSS, you may also have come through these typical code structures.

```
nav {
    height: 40px;
    width: 100%;
    background: #455868;
    border-bottom: 2px solid #283744;
}
nav li {
    width: 600px;
    height: 40px;
}
nav li a {
    color: #fff;
    line-height: 40px;
    text-shadow: 1px 1px 0px #283744;
}
```

In plain CSS, we select child elements by first targeting the parent in every rule set, which is considerably redundant if we follow the "best practices" principle.

In LESS CSS, we can simplify the above rule sets by nesting the child elements inside the parents, as follows;

```
nav {
    height: 40px;
    width: 100%;
    background: #455868;
    border-bottom: 2px solid #283744;
    li {
        width: 600px;
        height: 40px;
        a {
            color: #fff;
            line-height: 40px;
            text-shadow: 1px 1px 0px #283744;
        }
    }
}
```

You can also assign pseudo-classes, like the :hover, to the selector using ampersand (&) symbol. Let's say we want to add :hover to the anchor tag above, we can write it this way:

```
a {
    color: #fff;
    line-height: 40px;
    text-shadow: 1px 1px 0px #283744;
    &:hover {
        background-color: #000;
        color: #fff;
    }
}
```

## Operation

We can also perform Operation in LESS, we can do something like addition, subtraction, multiplication and division to number, color and variable in the style sheet.

Let's say we want the element B two times higher than element A, in that case, we can write it this way;

```
@height: 100px

.element-A {
    height: @height;
}
.element-B {
    height: @height * 2;
}
```

As you can see above, we first store the value in the @height variable, then assign the value to element A.

In the element B, rather than calculate the height ourselves, we can simply multiply the height by 2 using the asterisk operator (*), so whenever we change the value in the @height variable, element B will always have twice the height.

Furthermore, we have also shown you some advanced operation examples in our tutorial here: Designing A Slick Menu Navigation Bar.

## Scope

LESS applies the Scope concept, where variables will be inherited first from the local scope, and when it is not available locally it will search up a wider scope.

```
header {
    @color: black;
    background-color: @color;
    nav {
        @color: blue;
        background-color: @color;
        a {
            color: @color;
        }
    }
}
```

In the example above, the header has a black background color, but nav's background color will be blue as it has @color variable in its local scope, while the a will also have blue that is inherited from its nearer parent, nav.

## Conclusion

LESS is only one of a few solutions of CSS pre-processor, you can also try SASS and the Stylus, which we also hope to cover in our future posts.

Ultimately, we hope this post can give you a basic undestanding on how we can write CSS in a better way using LESS. If this is your first time using LESS, you maybe feel a little clumsy, but as you try it more often, it will surely become a lot easier.

**Note:** http://www.hongkiat.com/blog/less-basic/