CSS PREPROCESSORS - LESS

Tehran Institute of Technology

Zahra Mansoori
z.mansoori@gmail.com
http://tarsimm.com

Contents

1. Introduction

A **CSS preprocessor** is a program that lets you generate CSS from the preprocessor's own unique syntax. There are many CSS preprocessors to choose from, however most CSS preprocessors will add some features that don't exist in pure CSS, such as **mixin**, **nesting selector**, **inheritance selector**, and so on. These features make the CSS structure more readable and easier to maintain.

To use a CSS preprocessor, you must

1. install a CSS compiler on your web server; Or
2. Use the CSS preprocessor to compile on the development environment, and then upload compiled CSS file to the web server

Syntactically Awesome Stylesheets (Sass) and **Leaner CSS (LESS)** are both CSS preprocessors. They are special stylesheet extensions that make designing easier and more efficient. Both **Sass** and **LESS** compile into CSS stylesheets so that browsers can read them. This is a necessary step because modern browsers cannot read.

1.1. Less or Sass?

Both preprocessors share some of the same properties: Both SASS and LESS allow for the use of mixins and variables. One difference, though, is that **SASS** is based on **Ruby**, while **LESS** uses **JavaScript**. But even this doesn't give either of the preprocessors an advantage over the others.

Hint: Ruby is a scripting language built from the ground up for use in front end and back end web development and similar applications. It is a robust, dynamically typed, object-oriented language with syntax so easy to understand and high-level that it is considered as close as you can get to coding in English.

The real differences are found instead in the **logical functions**: LESS gives users the opportunity to only activate mixins when specific situations occur. This is a helpful feature, but it represents the extent of logical links in LESS. SASS, on the other hand, offers loops and case distinctions as known from programming languages.

1.2. Which application do we use?

As it is required for us to use an application which support Less, we chose Crunch2 App as a compiler for our projects.

Get it from https://getcrunch.co/

Or:

https://github.com/Crunch/Crunch-2/releases/download/v2.5.1/CrunchSetup.exe

2. Less Introduction

LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for website. LESS is a dynamic style sheet language that extends the capability of CSS.

2.1. Less — History

LESS was designed by Alexis Sellier in 2009. LESS is an open-source. The first version of LESS was written in **Ruby**; in the later versions, the use of Ruby was replaced by **JavaScript**.

2.2. Less — Features

- Cleaner and more readable code can be written in an organized way
- We can define styles and it can be reused throughout the code
- LESS is based on **JavaScript** and is a super set of CSS
- LESS is an agile tool that sorts out the problem of code redundancy

2.3. LESS — NESTED RULES

It is a group of CSS properties which allows using properties of one class into another class and includes the class name as its properties. In LESS, you can declare mixin in the same way as CSS style using class or id selector. It can store multiple values and can be reused in the code whenever necessary.

Example. 1: The following example demonstrates the use of nested rules in the LESS file

```
<html>
	<head>
		<title>Nested Rules</title>
		<link rel="stylesheet" type="text/css"
		href="style.css" />
	</head>
	<body>
		<div class="container">
			<h1>First Heading</h1>
			<p>LESS is a dynamic style sheet language
			that extends the capability of CSS.</p>
			<div class="myclass">
				<h1>Second Heading</h1>
				<p>LESS enables customizable, manageable
				and reusable style sheet for web
				site.</p>
			</div>
		</div>
	</body>
</html>
```

style.less

```
.container{
	h1{
		font-size: 25px;
		color:#E45456;
	}
	p{
		font-size: 25px;
		color:#3C7949;
	}
	.myclass{
		h1{
			font-size: 25px;
			color:#E45456;
		}
		p{
			font-size: 25px;
```

```
                    color:#3C7949;
                }
            }
        }
```

Output

style.css

```
        .container h1 {
                font-size: 25px;
                color: #E45456;
        }

        .container p {
                font-size: 25px;
                color: #3C7949;
        }
        .container .myclass h1 {
                font-size: 25px;
                color: #E45456;
        }
        .container .myclass p {
                font-size: 25px;
                color: #3C7949;
        }
```

2.4. LESS — NESTED DIRECTIVES & BUBBLING

You can nest the directives such as *media* and *keyframe* in the same manner, the way you nest the selectors. You can place the directive on top and its relative elements will not be changed inside its rule set. This is known as the bubbling process.

Example. 2: Nested directives and bubbling

```
<html>
	<head>
		<title>Nested Directives</title>
		<link rel="stylesheet" type="text/css"
		href="style.css" />
	</head>
	<body>
		<h1>Example using Nested Directives</h1>
		<p class="myclass">LESS enables customizable,
		manageable and reusable style sheet for web
		site.</p>
	</body>
</html>
```

style.less

```
.myclass {
	@media screen {
		color: blue;
		@media (min-width: 1024px) {
			color: green;
		}
	}

	@media print {
		color: black;
	}
}
```

Output

style.css

```
@media screen {
	.myclass {
		color: blue;
	}
}
```

```
@media screen and (min-width: 1024px) {
      .myclass {
            color: green;
      }
}
@media print {
      .myclass {
            color: black;
      }
}
```

2.5. LESS — OPERATIONS

LESS provides support for some arithmetical operations such as plus (+), minus (-), multiplication (*) and division (/) and they can operate on any number, color or variable. Operations save lot of time when you are using variables and you feel like working on simple mathematics.

Example. 3: Use of operations

```
<html>
      <head>
            <title>Less Operations</title>
            <link rel="stylesheet" type="text/css"
            href="style.css" />
      </head>
      <body>
            <h1>Example using Operations</h1>
            <p class="myclass">LESS enables customizable,
            manageable and reusable style sheet for web
            site.</p>
      </body>
</html>
```

style.less

```
@fontSize: 10px;
.myclass {
      font-size: @fontSize * 2;
      color:green;
}
```

Output

style.css

```
.myclass {
	font-size: 20px;
	color: green;
}
```

2.6. LESS — ESCAPING

It builds selectors dynamically and uses property or variable value as arbitrary string.

Example. 4: Use of escaping

```
<html>
	<head>
		<title>Less Escaping</title>
		<link rel="stylesheet" type="text/css"
		href="style.css" />
	</head>
	<body>
		<h1>Example using Escaping</h1>
		<p>LESS enables customizable, manageable and
		reusable style sheet for web site.</p>
	</body>
</html>
```

style.less

```
p {
	color: ~"green";
}
```

Output

style.css

```
p {
	color: green;
}
```

2.7. LESS — FUNCTIONS

LESS maps JavaScript code with manipulation of values and uses predefined functions to manipulate HTML elements aspects in the style sheet. It provides several functions to manipulate colors such as round function, floor function, ceil function, percentage function, etc.

Example. 5: Use of functions

```
<html>
      <head>
            <title>Less Functions</title>
            <link rel="stylesheet" type="text/css"
            href="style.css" />
      </head>
      <body>
            <h1>Example using Functions</h1>
            <p class="mycolor">LESS enables customizable,
            manageable and reusable style sheet for web
            site.</p>
      </body>
</html>
```

style.less

```
@color: #FF8000;
@width:1.0;

.mycolor{
      color: @color;
      width: percentage(@width);
}
```

Output

style.css

```
.mycolor {
      color: #FF8000;
      width: 100%;
}
```

Refer to:

https://lesscss.org/

and download Less.js . Using Less.js in the browser is the easiest way to get started and convenient for developing with Less, but in production, when performance and reliability is important, we recommend pre-compiling using Node.js or one of the many third party tools available.

To start off, link your .less stylesheets with the rel attribute set to "stylesheet/less":

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

Next, download less.js and include it in a <script></script> tag in the <head> element of your page:

```
<script src="less.js" type="text/javascript"></script>
```

Or use CDN:

```
<script src="//cdn.jsdelivr.net/npm/less@3.13" ></script>
```

Hint: You have to run this script under apache server or either a server which compile JavaScript. Nevertheless, this script won't work

2.7.1. Logical Functions – if

Returns one of two values depending on a condition.

Less file

```
@some: foo;

div {
margin: if((2 > 1), 0, 3px);
   color:  if((iscolor(@some)), @some,
black);
}
```

CSS File

```
div {
   margin: 0;
   color:  black;
}
```

2.7.2. Boolean

Evaluates to true or false. You can "store" a boolean test for later evaluation in a guard or **if**.

Less file	CSS File

@bg: black;
@bg-light: boolean(luma(@bg) > 50%);
 	
div {
background: @bg;
color: if(@bg-light, black, white);
}
	div {
	background: black;
	color: white;
	}

2.7.3. String Functions - escape

Applies URL-encoding to special characters found in the input string.

- These characters are not encoded: `,`, `/`, `?`, `@`, `&`, `+`, `'`, `~`, `!` and `$`.
- Most common encoded characters are: `\<space\>`, `#`, `^`, `(`, `)`, `{`, `}`, `|`, `:`, `>`, `<`, `;`, `]`, `[` and `=`.

Less file	CSS File
escape('a=1')	a%3D1

2.7.4. String Functions - e

String escaping. It expects string as a parameter and return its content as is, but without quotes. It can be used to output CSS value which is either not valid CSS syntax, or uses proprietary syntax which Less doesn't recognize.

Less file	CSS File
@mscode: "ms:alwaysHasItsOwnSyntax.For.Stuff()" filter: e(@mscode);	filter: ms:alwaysHasItsOwnSyntax.For.Stuff();

For more information, refer to http://lesscss.org/functions/

2.8. LESS — SCOPE

Variable scope specifies the place of the available variable. The variables will be searched from the local scope and if they are not available, then compiler will search from the parent scope.

Example. 6: Use of scoping

```
<html>
	<head>
		<title>Less Scope</title>
		<link rel="stylesheet" type="text/css"
		href="style.css" />
	</head>
	<body>
		<h1>Example using Scope</h1>
		<p class="myclass">LESS enables customizable,
		manageable and reusable style sheet for web
		site.</p>
	</body>
</html>
```

style.less

```
@var: @a;
@a: 15px;



.myclass {
	font-size: @var;
	@a:20px;
	color: green;
	}
```

Output

style.css

```
.myclass {
	font-size: 20px;
	color: green;
	}
```

2.9. LESS — COMMENTS

Comments make the code clear and understandable for the users. You can use both the block style and the inline comments in the code, but when you compile the LESS code, the single line comments will not appear in the CSS file.

Example. 7: Use of comments

```
<html>
      <head>
            <title>Less Comments</title>
            <link rel="stylesheet" type="text/css"
            href="style.css" />
      </head>
      <body>
            <h1>Example using Comments</h1>
            <p class="myclass">LESS enables customizable,
            manageable and reusable style sheet for web
            site.</p>
            <p class="myclass1">It allows reusing CSS code and
            writing LESS code with same semantics.</p>
      </body>
</html>
```

style.less

```
/* It displays the
green color! */
.myclass{
      color: green;
      }
// It displays the blue color
.myclass1{
      color: red;
      }
```

Output

style.css

```
/* It displays the
green color! */
.myclass {
      color: green;
      }
.myclass1 {
      color: red;
      }
```

2.10. LESS — IMPORTING

It is used to import the contents of the LESS or CSS files.

Example. 7: Use of comments

```
<html>
      <head>
            <title>Less Importing</title>
            <link rel="stylesheet" type="text/css"
            href="style.css"/>
      </head>
      <body>
            <h1>Example using Importing</h1>
            <p class="myclass">LESS enables customizable,
            manageable and reusable style sheet for web
            site.</p>
            <p class="myclass1">It allows reusing CSS code and
            writing LESS code with same semantics.</p>
            <p class="myclass2">LESS supports creating cleaner,
            cross-browser friendly CSS faster and easier.</p>
      </body>
</html>
```

myfile.less

```
.myclass{
      color: #FF8000;
}
.myclass1{
      color: #5882FA;
}
```

style.less

```
@import "myfile.less";
.myclass2
{
color: #FF0000;
}
```

Output

style.css

```
.myclass {
      color: #FF8000;
}
.myclass1 {
      color: #5882FA;
}
.myclass2 {
      color: #FF0000;
}
```

2.11. LESS — MIXIN

LESS always tries to keep its syntax and features so that it closely mimics CSS. Hence a mixin in LESS is declared in the same way as we declare a CSS style rule using a class or an ID selector. Both of the following examples are valid LESS mixins:

Less file

```
.border {
        border: 1px solid #ccc;
}
#shadow {
        box-shadow: 3px 3px 3px 0
    rgba(0,0,0,0.2);
}
.header {
        .border;
        #shadow;
        color: #000;
}
```

CSS File

```
.border {
        border: 1px solid #ccc;
}
#shadow {
        box-shadow: 3px 3px 3px 0
    rgba(0, 0, 0, 0.2);
}
.header {
        border: 1px solid #ccc;
        box-shadow: 3px 3px 3px 0
    rgba(0, 0, 0, 0.2);
        color: #000;
}
```