What this book will do for you...

- Integrate Less into your projects to boost efficiency
- Spend less time debugging
- Compile Less code into readable and maintainable CSS
- Write reusable and portable code and avoid duplication
- Make use of prebuilt and proven code
- Reduce the development and maintenance time of your projects
- Set up a development environment with Grunt

Inside the Cookbook...

- A straightforward and easy-to-follow format
- A selection of the most important tasks and problems
- Carefully organized instructions for solving the problem efficiently
- Clear explanations of what you did
- Apply the solution to other situations

Quick answers to common problems

Less Web Development Cookbook

Over 110 practical recipes to help you write leaner, more efficient CSS code

In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 1 “Getting to Grips with the Basics of Less”
- A synopsis of the book’s content
- More information on Less Web Development Cookbook

About the Authors

**Bass Jobsen** has been programming for the Web since 1995, covering everything from C to PHP, and is always on the hunt to find the most accessible interfaces. Based in Orthen, the Netherlands, he has also written *Less Web Development Essentials, Packt Publishing*, which is a fast-paced tutorial that covers the fundamentals of Less (Leaner CSS) when used in web development.

Bass uses Less in his daily job for web design tasks, WordPress theme development, and other Twitter Bootstrap apps.

He is always happy to help those with questions (http://stackoverflow.com/users/1596547/bass-jobsen), and he writes a blog you can find at http://bassjobsen.weblogs.fm/.

Also, check out his Bootstrap WordPress starters theme (JBST) and other projects at GitHub (https://github.com/bassjobsen).

This book is for Colinda, Kiki, Dries, Wolf, and Leny.

Writing this book wasn't possible without the support of my family, Caroliene, and the people of Vivent. Richard Harvey is a patient and excellent motivator and critical reader. Akashdeep Kundu helped me to dot the i's and cross the t's. I'd also like to thank the reviewers of this book, Dave Poon, Steve Workman, and Fahad Heylaal, for their critical and valuable suggestions that made this book even better.

Last but not least, I should not forget to thank the Less core Team: Alexis Sellier (@cloudhead), Jon Schlinkert (@jonschlinkert), Luke Page (@lukeapage), Marcus Bointon (@Synchro), Mária Jurčovičová (@sommeri), Matthew Dean (@matthew-dean), Max Mikhailov (@seven-phases-max), and all the other contributors who have made coding Less possible in the first place.
Amin Meyghani is a designer and developer currently working at HD MADE (http://hdmade.com/), making automation tools, websites, and apps. He is also a lead developer at Flitti (http://flitti.com/), leading the team to make next-generation gamification apps. In addition to arts and technology, Amin has always been passionate about teaching. He takes advantage of every opportunity to share his knowledge with the world through books, blogs, or videos. You can find his works and blogs at http://meyghani.com/. When Amin is not coding, he is either enjoying Persian food or mastering his Persian calligraphy techniques.

I would like to thank my family for always supporting me and filling my life with love and hope. I owe them this book, as they have always been there for me even in the most difficult times.
CSS has dramatically changed since its very first emergence, and it is continuing to evolve. In particular, the emergence of CSS3 has added many new features to CSS, including gradients and animations. Along with this are many new opportunities to build websites using only CSS and HTML. Developers are no longer dependent on techniques such as Flash and other tricks to build interactive and fancy websites.

CSS3 has played an integral role in building responsive websites, where CSS media queries have made it possible to apply some styles dependent on the width of the browser's viewport only.

Despite this improvement, CSS is inherently, at its core, a simple style sheet language that lacks some fundamental programming features such as variables, functions, and operators. The need for more maintainable CSS, especially with the explosion of complex web apps, has made CSS preprocessors such as Less a necessity in enabling us to write more readable and manageable versions without breaking cross-browser compatibilities.

Although Less cannot magically change CSS, it certainly provides us with the tools to help structure, modularize, debug, and maintain small or large CSS projects more easily. By extending CSS with variables, functions, and mixins; nesting CSS selectors; and allowing you to follow the don't repeat yourself (DRY) principle of software programming, Less behaves more like a programming language in a way that CSS never was. Despite some of the programming characteristics of Less, you should not be put off by this; by being built as a superset of CSS, its features are implemented in the CSS way and it follows W3C standards where possible. Designers and developers who are familiar with CSS will find coding in Less very natural. Because Less fixes these shortcomings of CSS, the best time to start using Less is now!

In this book, you are going to explore the Less preprocessor, most of its core, and some of its less frequently used features. Through these very easy-to-follow and practical recipes, you will learn how to write more maintainable and scalable CSS. You will explore making components and structures through reusable mixins and extends. We will also learn about frameworks that are based on Less, exploring their features and how they can be seamlessly integrated into your own projects. In addition, you will learn how to use prebuilt mixin libraries for your current or upcoming projects. Finally, you will look at debugging techniques that have been available for other preprocessors and are now available to Less through source maps. By the end of this book, you will have an extended knowledge and a good understanding of the power of Less, its libraries, and the important features it has to offer to make writing your CSS more natural, productive, and intuitive.
What This Book Covers

Chapter 1, Getting to Grips with the Basics of Less, shows you how to install the Less compiler for client- and server-side usage. After the installation, you will be shown how to make use of the basic features of Less: using variables, mixins, operations, built-in functions, and namespaces; how to nest your rules will also be on the menu here!

Chapter 2, Debugging and Documenting Your Less Code, shows you how to debug your Less code using your CSS source maps and browser developer tools. You will also be introduced to style guides and learn how to properly comment your code.

Chapter 3, Using Variables and Mixins, covers the advanced usage of variables and mixins in Less. After reading this chapter, you will know how to use variables to create reusable Less code and use mixins to make your CSS properties interactive.

Chapter 4, Leveraging the Less Built-in Functions, explains the different types of built-in functions of Less. You will find examples of each type of function, including functions for color manipulation and mathematical operations.

Chapter 5, Extending and Referencing, shows you how to extend and reference selectors and properties to help you write better CSS and reduce the size of the compiled CSS code. You will learn to change the order of selectors and merge them.

Chapter 6, Advanced Less Coding, walks you through the process of parameterized mixins and shows you how to use guards. A guard can be used with as if-else statements and make it possible to construct interactive loops in Less.

Chapter 7, Leveraging Libraries with Prebuilt Mixins, explains how to install and use the libraries of prebuilt mixins. You will explore different libraries such as Less Elements, Less hats, and Preboot to build background gradients, grids, and animations in Less, among others.

Chapter 8, Building a Layout with Less, takes you through the process of creating a complete website layout with Less. The layout will be built with a responsive and semantic grid and will include a vertical menu. Finally, you will also learn how to use iconic fonts with Less.
Chapter 9, *Using Bootstrap with Less*, shows you how to customize Bootstrap and its components using Bootstrap's Less source files. You will also learn how to use Bootstrap's mixins to make semantic and reusable layouts and components.

Chapter 10, *Less and WordPress*, shows you how to use Less when theming your WordPress site. This includes examples of the Roots.io, SemanticUI, and JBST WordPress themes. You will be shown how to customize the WooCommerce plugins with Less along with integrating Less into your other WordPress themes and plugins.

Chapter 11, *Compiling Less Real Time for Development Using Grunt*, shows you how to set up a Less compiler using Grunt for real-time compilation. It will also show you how to use several Node modules for your Less development. By the end of this chapter, you will be very comfortable setting up a development environment with Grunt and its plugins.
Getting to Grips with the Basics of Less

In this chapter, we will cover the following topics:

- Downloading, installing, and integrating less.js
- Installing the lessc compiler with npm
- Using less.js with Rhino
- Declaring variables with Less for commonly used values
- Setting the properties of CSS styles with mixins
- Writing more intuitive code and making inheritance clear with nested rules
- Creating complex relationships between properties
- Using the built-in functions of Less
- Using namespaces to make your code reusable and portable

Introduction

Leaner CSS (Less) is a preprocessor for CSS code. This chapter will guide you through the installation of Less. It can be used on the command line via npm (or Rhino) or downloaded as a script file for a web browser. Other third-party compilers are available too.

Although client-side compiling is not suitable for production, it is very useful to develop and test your code. A client-side compiler will run in any modern browser and show you the effect of your coding in real time. On the other hand, the server-side-compiled CSS code can be minified and used for production. Note that client-side compiling doesn't save the output and compiles your code again after each browser reload, while the output of the server-side compiler will be saved in a static CSS file.
Getting to Grips with the Basics of Less

You will also see that Less, in contrast to CSS, is a programming language for writing CSS more efficiently. It adds built-in functions, variables, and mixins with a lot more to offer to CSS, which helps you to meet the Don't repeat yourself (DRY) principle of software programming and reuse your code. Variables enable you to define the commonly used values only once, and mixins create the reusable blocks of code. You will work more effectively and find that you will spend less time on debugging and maintaining your projects.

Less extends the CSS language, which also means that valid CSS code is valid Less code. Whoever is familiar with CSS will find that the process of learning Less has a flat learning curve and is very intuitive.

After installing Less, the other recipes in this chapter will show you its basic features and how to use them to write a better, reusable, and more maintainable CSS code.

**Downloading, installing, and integrating less.js**

The client-side compiler less.js can be downloaded from [http://lesscss.org/](http://lesscss.org/). You can use less.js in the browser, which is a great tool to get you started with Less, although it should only be used for development. For production usage, you should use precompiling. Precompiling with the Node.js compiler will be discussed in the Installing the lessc compiler with npm recipe.

**Getting ready**

You can download the latest version of less.js from [http://lesscss.org/](http://lesscss.org/) and copy this file into your working directory. You will also have to create the index.html and project.less files in the working directory. You can edit these files with any text editor of your choice.

You will have the following folder and file structure:

```
|-- index.html
|-- less
   |-- project.less
   |-- less.js
```

You will also need a modern web browser to inspect the results of your work.
It is not necessary to have a web server running. Navigating to the index.html file on your hard drive with your browser will be enough. However, this won’t work for all browsers, so use Mozilla Firefox to be sure when you do not have a web server running. The examples in this book use http://localhost/map/ and can be replaced with the path similar to file:///map/ or c:\map\, depending on your situation.

### How to do it...

1. To start the process, you will have to edit your index.html file. The index.html file should contain a valid HTML5 code and have references to the project.less and less.js files. After you edit it, the HTML file will look as follows:

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet/less" type="text/css" href="project.less">
    <script src="less.js" type="text/javascript"></script>
  </head>
  <body>
    <header>the header</header>
    <section>this is a paragraph</section>
    <footer>the footer</footer>
  </body>
</html>
```

2. The HTML5 code from the preceding step contains a header, section, and footer. Now you can use Less to style these elements. Enter the following code into the project.less file to give each element a different font color:

```less
header {
  color: blue;
}
section {
  color: green;
}
footer {
  color: purple;
}
```

3. Finally, you can inspect the result of your actions by opening the index.html file in your web browser.
Getting to Grips with the Basics of Less

How it works...

The less.js compiler compiles the Less code in the project.less file linked with the following HTML code:

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

Note that without setting the rel="stylesheet/less" attribute, the compiler does not recognize your code.

The reference to the less.js compiler should be included after the reference to the project.less file in the preceding code as follows:

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

Other Less files can be imported into project.less with the @import directive. All imports are compiled into CSS code. Also note that when using the server-side compiler, all of the compiled CSS code will be saved to the same file. The preceding code differs from the situation of linking more than one .less style sheet. When linking multiple .less style sheets, each file will be compiled independently and will not use variables and mixins defined in the other files.

The compiled CSS code will be injected into the HTML document and style your HTML elements according to normal CSS rules. When using Firefox's or Google Chrome's developer tools to inspect your source, you will find the compiled code as follows:

```html
<STYLE type="text/css">
/*<style type="text/css">
  @import "project.less";
  /*<style type="text/css">
</STYLE>
```

The less.js file also has a watch function that checks your files for changes and reloads your browser views automatically when changes are found. It is pretty simple to use—add #!watch after the URL you want to open, which in this case means appending #!watch after index.html, and then reload the browser window.
There's more...

You can configure the less.js compiler by adding a `less = {};` JavaScript object to your code using the following code:

```html
<link rel="stylesheet/less" type="text/css"
     href="less/styles.less" />
<script type="text/javascript">less = { env: 'development' }
</script>
<script src="less.js" type="text/javascript"></script>
```

In the preceding code, `less` is a global object used to parse the `env: 'development'` settings to `less.js`. Please refer to http://lesscss.org/#client-side-usage-browser-options to learn more about the settings that can be used with the `less.js` compiler.

Alternatively, these options can be set as data attributes on the `script` and `link` tags, as can be seen in the following example code from the Less website:

```html
<script src="less.js" data-poll="1000"
       data-relative-urls="false"></script>
<link data-dump-line-numbers="all" data-global-vars='{ myvar: "#ddffee", mystr: "\"quoted\""}' rel="stylesheet/less"
      type="text/css" href="less/styles.less"
</script>
```

In this recipe, a local copy of the `less.js` compiler was used. Alternatively, you can also load the `less.js` compiler from content delivery network (CDN) or build it with Bower. To load `less.js` from CDN, you will have to add the following code to your `index.html` file:

```html
<script src="//cdnjs.cloudflare.com/ajax/libs/less.js/2.x.x/less.min.js"></script>
```

If you aren't aware, Bower is a package manager for the Web. You can install Bower by running the following command in your console:

```
npm install bower
```

You can then run the following command to build `less.js` with Bower:

```
bower install less
```

See also

- More information about Bower can be found at http://bower.io/
Installing the lessc compiler with npm

For server-side compilation, Less comes with a command-line compiler for Node.js. The node package manager (npm) can be used to install the Less command-line compiler.

Node is a platform built on Chrome’s JavaScript runtime called V8, allowing you to easily create fast and scalable network applications.

Getting ready

If you have not installed Node.js and npm on your system yet, you will have to do this first. You can do this by following these steps:

1. Download the Node.js source code or a prebuilt installer for your platform from http://nodejs.org/download/.
2. Install Node.js, which includes npm, on your system.

In the Installing Node and Grunt recipe in Chapter 11, Compiling Less Real Time for Development Using Grunt, you can read about installing Node.js and npm on your system in more detail. After installing npm, you can simply run the following command:

```
npm install --global less
```

How to do it...

1. For this recipe, you will first need to create a simple Less file and save this file, which for instance might be example.less. You can try the following code in your example file:

```
@color: red;
.paint() {
  color: @color;
}
p {
  .paint();
}
```

2. After creating the Less file in the preceding format, you will need to save your file (which may be example.less or whatever filename you have chosen). If you have chosen example.less, you can run the following command in your command prompt:

```
lessc example.less
```
3. After running the `lessc` command, you will see it output the following CSS code in the console:

```css
p {
  color: #ff0000;
}
```

**How it works…**

If you are new to Less, the example Less code used inside `example.less` may contain some syntax that is completely alien to you. The code defines a `@color` variable and a `paint()` mixin. The *Declaring variables with Less for commonly used values* recipe explains the basics of variables in Less, while the *Setting the properties of CSS styles with mixins* recipe does the same for mixins.

By default, the `lessc` compiler outputs to `stdout`. You can redirect the output to a CSS file with the following command:

```
lessc example.less > example.css
```

Running the `lessc` compiler without any parameters will give you a list of options for the compiler.

You can use the `-x` option to compress your output as follows:

```
lessc -x example.less > example.css
```

In a similar manner, you can use either the `--clean-css` option for a more involved minification, or the `--source-map` option to create a v3 CSS source map. In the *Using CSS source maps to debug your code* recipe in Chapter 2, *Debugging and Documenting your Less Code*, you can read more about CSS source maps and Less. Note that in version 2 of Less, the `--clean-css` option has been moved into a plugin. The usage is similar: just install the plugin (`npm install -g less-plugin-clean-css`), then make use of the `--clean-css` argument.

**There's more...**

There are many other third-party compilers for Less with a compressive list available at [http://lesscss.org/usage](http://lesscss.org/usage).

With `grunt-contrib-less`, you can compile your code with Grunt. For Gulp, you can use `gulp-less`. The *Compiling style guides with Grunt* recipe in Chapter 11, *Compiling Less Real Time for Development Using Grunt*, shows you how to build a development workflow with the Grunt task runner.
Getting to Grips with the Basics of Less

In this recipe, you read about Grunt and Gulp, which are JavaScript task runners or build systems. Comparing with Grunt's build system, Gulp's build system is relatively new. Gulp uses streams and code over configuration, which makes it more simple and intuitive.

See also

- In *Chapter 11, Compiling Less Real Time for Development Using Grunt*, you can read all you want to know about Grunt
- To read more about Gulp, you can visit http://gulpjs.com/
- The Gulp Less plugin can be found at https://github.com/plus3network/gulp-less

Using less.js with Rhino

Less also runs inside Rhino, which is an open source implementation of JavaScript written entirely in Java. It is typically embedded into Java applications to provide scripting to end users. Rhino enables you to use the original less.js distribution in a pure JVM environment.

Getting ready

To use less.js inside Rhino, you will have to download and install Rhino from the following links:

- Download and install the latest version of Rhino from https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino/Download_Rhino
- Download the Rhino version of less-rhino-x.x.x.js and lessc-rhino-x.x.x.js from https://github.com/less/less.js/tree/master/dist

How to do it...

1. Open your text editor and create a file named example.less. The example.less file can contain, for instance, the following code:
   ```
   @base-color: red;
   h1 {
       color: @base-color;
   }
   ```

2. Now you can run the following command in your command prompt:
   ```
   java -jar js.jar -f less-rhino-1.7.0.js lessc-rhino-1.7.0.js example.less
   ```
3. The preceding command should output the following lines of CSS code:

```css
h1 {
  color: #ff0000;
}
```

**How it works...**

Rhino enables Java to run the JavaScript code, while `js.jar` runs the Less compiler and generates the CSS output.

To write the output of a file, you will have to append the filename of the CSS files to the list of commands, as follows:

```bash
java -jar js.jar -f less-rhino-1.7.0.js lessc-rhino-1.7.0.js example.less example.css
```

You can also add options for the compiler. You can add the `-x` option to compress the output as follows:

```bash
java -jar js.jar -f less-rhino-1.7.0.js lessc-rhino-1.7.0.js -x example.less
```

The preceding command will then output the following line of CSS code:

```css
h1{color:#f00}
```

**There's more...**

A Less compiler for Java has been built with Rhino. You can find out more information about this Less compiler for Java along with how to download it at https://github.com/marceloverdijk/lesscss-java.

**Declaring variables with Less for commonly used values**

Less allows you to use variables. You can assign a variable a value, which will be called a declaration. After a variable is declared, you can use the variable anywhere in your code to reference its value. Variables allow you to specify widely used values in a single place and then reuse them throughout your code. Defining once also means you have to edit it once when you want to change its value.
Getting ready

Open your text editor and create a file named example.less. Variables will start with @ and will have a name with examples, including @color, @size, and @tree. To write the name, you are allowed to use any alphanumeric characters, underscores, and dashes. Using this as an elaborate example, @this-is-variable-name-with-35-chars is a valid variable name.

How to do it...

1. Start with creating a simple HTML5 file named index.html, as follows:

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Use variables in Less</title>
    <link rel="stylesheet/less" type="text/css" href="example.less">
    <script src="less.js" type="text/javascript"></script>
</head>
<body>
    <h1>Color your page with variables</h1>
    <p>Hello Less</p>
    <button>Click here</button>
</body>
</html>
```

2. You then need to create the example.less file, which should contain the following code:

```less
@base-color: red;
h1 {
    color: @base-color;
}
p{
    color: @base-color;
}
button {
    color: @base-color;
}
```
3. After the first two steps, you will end up with the following folder and file structure:

```
-index.html
- less
-- example.less
- less.js
```

4. After creating the files as described in the preceding steps, you can open index.html in your browser.

5. Now, change the first line of code `@base-color: red;` to `@base-color: green;` and reload your browser.

**How it works...**

As you can now see, changing the font color of the h1, p, and button text is easy as you change `@base-color` only once. The only thing you need to do is change the single line of the code: `@base-color: red;`. In the *Downloading, installing, and integrating less.js* recipe, you can read how to use the `watch` function of less.js to reload your browser automatically after changing and saving the `example.less` file.

Variables in Less are defined as the equivalent to statics in other programming languages. You assign a value to a variable once and use it everywhere in your code. To think of it in another way, this is like defining the value of the gravitational constant (for the force of gravity) or pi in your code. Both these values become constants once they are declared and so do not change at runtime. In fact, you can still change or redeclare them in Less, as explained in the There's more... section of this recipe.

You can assign any valid Less (or CSS) property value to a variable. Valid property values include the numbers, strings, lists, CSV lists, and escaped values. Strings and numbers can be used together to define values with units. For instance, the following code will show you a declaration for a length in pixels:

```
@length: 100px;
```

Other examples of valid variable declarations can be found in the following code:

```
@color: red;
@list: a b c d;
@csv-list: a, b, c, d;
@escaped-value: ~"dark@{color}";
```
Getting to Grips with the Basics of Less

**There's more...**

Less uses the **last declaration wins** and **lazy loading** rules, which play an important role and make redeclaration of a variable suitable for customization.

**See also**

- You can read more about the usages of redeclaration variables for customization in the Redeclaring variables based on lazy loading recipe in Chapter 3, Using Variables and Mixins

**Setting the properties of CSS styles with mixins**

In Less, mixins hold a set of properties that can be reused for different rulesets. The properties of a mixin are included in the ruleset. The mixin itself does not generate output to the final CSS. They look like normal classes (or an ID ruleset, starting with #). Although they are optional, most mixin declarations end with parentheses, which prevent the mixins from compiling into the source. A mixin with parentheses is called a **parametric mixin**. You can read more about parametric mixins in the Using parametric mixins recipe in Chapter 3, Using Variables and Mixins.

**Getting ready**

Open your text editor and create a file named mixins.less. In this file, define a mixin for rounded corners, as follows:

```
.rounded-corners() {
  border-radius: 5px;
}
```

You will also need an index.html file containing some HTML elements to which you can give rounded corners.

**How to do it...**

1. You first need to create a valid HTML5 document named index.html with the following elements:

   ```html
   <header>the header</header>
   <p>this is a paragraph</p>
   <footer>the footer</footer>
   ```
Chapter 1

Make sure the head section of your index.html file also contains the following code:

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

Note that the preceding code references a Less file called `project.less` instead of `mixins.less`.

2. After creating the index.html file, you can start writing your Less code, which will give the HTML elements rounded corners. Since mixins can be reused, it will be a good practice to write them in a separated file, enabling you to import the mixins in your other projects too.

3. Now, create your `project.less` file. This file imports the mixin(s) from the `mixins.less` file using the following code:

```less
@import "mixins.less";
```

4. After creating the files, visit the `mixins.less` file. Here, write the following code:

```less
.rounded-corners() {
  border-radius: 5px;
}
```

5. Following this edit, you can give an HTML element rounded corners by adding the `rounded-corners()` mixin call to its property list. Finally, your `project.less` file will look as shown in the following code:

```less
@import "mixins.less";

@header-background-color: red;
@paragraph-background-color: orange;
@footer-background-color: green;

header {
  .rounded-corners();
  background-color: @header-background-color;
  color: contrast(@header-background-color);
}
p {
  .rounded-corners();
```
Getting to Grips with the Basics of Less

```less
background-color: @paragraph-background-color;
color: contrast(@paragraph-background-color);
}
footer {
  .rounded-corners();
  background-color: @footer-background-color;
color: contrast(@footer-background-color);
}
```

How it works...

Every element has a `background-color` and `color` property set to make the rounded corners visible and the fonts readable. The `color` property is set with the built-in `contrast` function. You can read more about the built-in functions in the Using the built-in functions of Less recipe. When you open the `index.html` file, it looks like the following screenshot:

```
the header
this is a paragraph
the footer
```

Less allows you to copy the properties of a class to another by simply adding the class to the property list. Consider the following example Less code:

```less
.class1
{
  property1: value1;
}
.class 2
{
  .class1
  property2: value2;
}
```

The preceding Less code will compile into the following CSS code:

```css
.class1 {
  property1: value1;
}
.class2 {
  property1: value1;
  property2: value2;
}
```
As you can see, the `property1` property is added to the `.class2` class, but `.class1` has also been compiled into the source. With parentheses, the `.class1` mixin is not compiled into the CSS source, so the following code will not be visible in the source:

```less
.class1() {
    property1: value1;
}
```

### There's more...

In the example code of this recipe, you set a `background-color` and `color` property for each element again. While using parametric mixins, as described in the *Using parametric mixins* recipe in *Chapter 3, Using Variables and Mixins*, you can write a second mixin to set these properties. The `roundedcorners()` mixin can be called from this particular mixin. The second mixin will then look like the following Less code:

```less
.roundedcorners() {
    .rounded-corners();
    background-color: @background-color;
    color: contrast(@background-color);
}
```

The `colored-and-rounded()` mixin can be added to the `mixins.less` file. Your `project.less` file will then look as follows:

```less
*import "mixins.less";*

@header-background-color: red;
@paragraph-background-color: orange;
@footer-background-color: green;

header {
    .colored-and-rounded();
}
p {
    .colored-and-rounded();
}
footer {
    .colored-and-rounded();
}
```
Getting to Grips with the Basics of Less

Writing more intuitive code and making inheritance clear with nested rules

HTML elements in the hierarchy of the Document Object Model (DOM) of HTML5 documents are nested while CSS, on the other hand, does not reflect this nested structure. Less makes nesting of CSS selectors possible. With the nested selectors being used, your code reflects the nested structure of HTML5.

Getting ready

To get started, you will need to create a valid HTML5 file, including some nested elements. Your HTML, for instance, may look like the following code:

```html
<section role="main">
  <h1>heading</h1>
  <p>some content</p>
</section>
```

You will also have to create an empty Less file named `project.less`. Make sure the head section of your HTML5 document also contains the following code:

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

How to do it...

In CSS, the section with the nested `h1` and `p` elements can, for instance, be styled with the following CSS code:

```css
section h1 {}
section p {}
```

However, with Less, you can style the same elements using the following Less code:

```less
section {
  h1 {}
  p{}
}
```
How it works...

In the preceding example, nesting the selector mimics the nested structure of your HTML code. Nesting makes the code intuitive and so much easier to read and maintain. Less's code will also be more concise than its corresponding CSS code. You should use nesting with care; nesting too much will break your CSS code after small changes in your HTML. You should not try to nest your complete HTML structure, but nesting will be very useful to assign pseudo classes, such as hover, to your elements.

Note that the nested selectors in Less still compile to un-nested selectors in CSS.

To see how this works, use the following Less code:

```less
section {
  h1 {font-size: 20em;}
  p { padding: 0 10px;}
}
```

The preceding code will compile into the following CSS code:

```css
section h1 { 
  font-size: 20em; 
}
section p { 
  padding: 0 10px; 
}
```

There's more...

Although nesting your selector can make your code more intuitive, it can equally break other things. For instance, considering object-oriented CSS (OOCSS) principles; these do not allow nesting of headings (h1 to h6). Headings are considered to be built-in objects in OOCSS and so their appearance should be consistent across an entire site.

See also

- You can read more about OOCSS at http://appendto.com/2014/04/oocss/
- Read the presentation by Nicole Sullivan used to introduce OOCSS at http://www.slideshare.net/stubbornella/object-oriented-css
Creating complex relationships between properties

Less supports basic arithmetic operations. These operations, such as division (/), can be applied to any number, variable, or even color. They can be used to create complex relationships between properties.

Getting ready

You will first need to create a Less file named `project.less` and a valid HTML5 document named `index.html`. You should make sure the head section of the `index.html` file contains the following lines of code:

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

How to do it...

1. First create an HTML structure in the `index.html` file as follows:
   ```html
   <div class="container">
   <section role="main">Content</section>
   <aside role="complementary">Side bar</aside>
   </div>
   ```

2. To set the width of the content and sidebar dependent on the width of the container, you can use the following Less code:
   ```less
   @basic-width: 800px;
   .container {
     width: @basic-width;
     section {
       width: @basic-width * 2/3;
       background-color:red;
       color:white;
       float:left;
     }
     aside {
       width: @basic-width * 1/3;
       background-color: black;
       color: white;
     }
   }
   ```
3. Now you can open the index.html file in your browser, which will look like the following screenshot:

Note that browsers can use different algorithms to round the pixel values when you assign them with decimal numbers. This phenomenon has also been described as Sub-Pixel problems in CSS. You can read more about these sub-pixel problems in CSS at http://ejohn.org/blog/sub-pixel-problems-in-css/.

How it works...

In Less, you can operate numbers, variables, and colors. The compiler understands colors and units. Take a look at the following Less code:

```less
@width: 50px;
@color: yellow;
p {
  width: @width * 50;
  color: @color + #111;
}
```

This will actually compile into the following CSS code:

```css
p {
  width: 2500px;
  color: #ffff11;
}
```

The Less compiler accepts different types of color definitions. In Less, you can use the hexadecimal notation for **red, green, and blue (RGB)** values, the RGB functional notation, or one of the 148 color names defined in CSS3. A complete overview of the color definition can be found at http://www.w3.org/TR/css3-color/ and http://lesscss.org/functions/#color-definition. When applying a basic operation on two or more colors, the compiler gives the result as a color even when different types of color definitions are used. As you can see, yellow + #111 compiles into #ffff11.

When multiplying 50px fifty times, the compiler automatically adds the px unit after the calculated result.
Getting to Grips with the Basics of Less

There's more...

In this recipe, you learned about some basic operations on colors. Less also has many built-in functions to define and manipulate colors. You can read more about Less's built-in color functions in Chapter 4, Leveraging the Less Built-in Functions.

See also

- At http://meyerweb.com/eric/css/colors/, you can find an overview of the 147 (or 148) color keywords
- Refer to http://meyerweb.com/eric/thoughts/2014/06/19/rebeccapurple/ for color names, which are defined in the CSS3 module along with their corresponding numeric equivalents

Using the built-in functions of Less

Less has many built-in functions that can be leveraged for others, transforming colors, manipulating strings, or even performing mathematical operations.

Getting ready

Create a valid HTML document named index.html and an empty project.less file. Make sure your index.html HTML5 document has the following lines of code in its head section:

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

How to do it...

This recipe will show you how to use the darken() and contrast() built-in functions. Perform the following steps:

1. Start this recipe by creating a simple HTML structure in the index.html file, shown as follows:
   ```html
   <div class="item color1">Text</div>
   <div class="item color2">Text</div>
   <div class="item color3">Text</div>
   <div class="item color4">Text</div>
   <div class="item color5">Text</div>
   ```
2. After creating the HTML page, add the following Less code to the `project.less` file:

```less
@start-color: white;
.color1 {
  background-color: @start-color;
  color: contrast(@start-color);
}
.color2 {
  @color: darken(@start-color, 25%);
  background-color: @color;
  color: contrast(@color);
}
.color3 {
  @color: darken(@start-color, 50%);
  background-color: @color;
  color: contrast(@color);
}
.color4 {
  @color: darken(@start-color, 75%);
  background-color: @color;
  color: contrast(@color);
}
.color5 {
  @color: darken(@start-color, 100%);
  background-color: @color;
  color: contrast(@color);
}
```

3. Now, open the `index.html` file in the browser and you will see the following output:

```
Text
Text
Text
Text
Text
```

**How it works...**

Both the `darken()` and `contrast()` functions return a color. The `darken()` function returns a darker variant of the input color, and `contrast()` returns black or white, based on the highest contrast with the input color.
The `darker()` function ensures that a color is readable against a background, which will be useful to meet web accessibility requirements too. The `contrast()` function compares the luma value (also called luminosity that represents the brightness in an image) of a color and not the lightness.

**There's more...**

The built-in functions of Less can be grouped based on their input type. Refer to the following functions:

- The **string** functions can be used to manipulate strings. The `replace` function, which replaces the text in a string, is an example of a string function.
- The **type** functions, which include functions such as `isnumber()` and `iscolor()`, return a Boolean value. The `iscolor()` function returns `true` for values such as `#ff0` or `red` and `false` for all other kinds of input types.
- The **list** functions operate on values. Both comma and space-separated lists are supported. The only two functions in the group are `extract()` and `length()`. The group of mathematical functions contain functions for all kinds of mathematical operations, such as `sin()`, `round()`, and `pow()`.
- Finally, there are four **groups of functions** that can be used with colors:
  - Color definition functions
  - Color channel functions
  - Color operations functions
  - Color blending functions

You will also have to note that the example code in this recipe did not meet the DRY principle of software programming. When using guards, as described in the *Building loops leveraging mixins and guards* recipe in Chapter 6, Advanced Less coding, you can solve this issue of code repetition. You can rewrite the Less code to the following code, which uses a guard:

```less
.shade(@color, @number) when (@number > 0) {
  .shade(@color, @number - 1);
  @darkcolor: darken(@color, (25% * (@number - 1)));
  .color@{number} {
    background-color: @darkcolor;
    color: contrast(@darkcolor);
  }
}
.shade(white, 5);
```
See also

A complete list of the built-in functions supported by Less can be found at http://lesscss.org/functions/.

Using namespaces to make your code reusable and portable

In programming languages, namespace is used to create a different scope for an object of a different origin. It prevents problems with such objects that have the same name. In Less, you can also create and use namespaces. They will help you to make your Less code more portable and reusable.

Getting ready

Open your text editor and create a file named project.less. If you don’t use the command-line compiler, you will also have to create a valid HTML5 document, including the following lines of code in the head section:

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

How to do it...

1. Create two mixins with the same name in the project.less file. You can, for instance, use the following code to create two mixins called mixin:

   ```less
   .mixin(){
     color: red;
   }
   .mixin(){
     color: blue;
   }
   e1 {
     mixin
   }
   ```

2. Now, compile the Less code you wrote in the project.less file and you will find that it will compile into the following code:

   ```less
   e1 {
     color:red;
     color:blue;
   }
   ```
3. After compiling your code, you can use the following Less code to wrap the first mixin in a namespace called #namespace, as follows:

```less
#namespace {
  .mixin() {
    color: red;
  }
}

.mixin() {
  color: blue;
}
```

4. Now the namespace `mixin` can be utilized with the following Less code:

```less
e1 {
  #namespace > mixin;
}
```

5. Finally, the Less code from the preceding step will compile into the following CSS code:

```css
e1 {
  color: red;
}
```

How it works...

The Less compiler doesn't throw an error for mixins with the same (or conflicting) names. The compiler compiles every matching mixin into the CSS code. You can read more about matching mixins in the Building loops leveraging mixins guards recipe in Chapter 6, Advanced Less coding.

Namespaces prevent conflicting names. In Less, a namespace starts with # and the code for it should be wrapped between accolades. A typical namespace will look as follows:

```less
#namespace { .mixin(){} }
```

Mixins inside a namespace can be called by adding the namespace before the mixin call, which you can see in the following code:

```less
#namespace > mixin;
```

You can also use the following code; it will generate the same result as you have obtained in the preceding code:

```less
#namespace mixin;
```

Note that the > sign in the preceding code is optional. The > sign has the same meaning as in CSS. In CSS, the > sign means a child (must be an immediate child, not any descendant).
Where to buy this book

You can buy Less Web Development Cookbook from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.