Less Web Development Essentials

Bass Jobsen

Chapter No. 1
"Improving Web Development with Less"
In this package, you will find:
A Biography of the author of the book
A preview chapter from the book, Chapter NO.1 "Improving Web Development with Less"
A synopsis of the book’s content
Information on where to buy this book

About the Author

Bass Jobsen has been programming for the Web since 1995, from C to PHP, always looking for the most accessible interfaces. He has a special interest in the process between a designer and programmer. He believes that interfaces should work independent of a device or browser. For these reasons, working with grids and meta languages in designs makes him happy. He always looks forward to new opportunities in the Semantic and Responsive Web.

He uses Less in his daily job for web design tasks and WordPress theme development as well as other Twitter Bootstrap apps.

He is always happy to help you. He can be reached at http://stackoverflow.com/users/1596547/bass-jobsen.

For More Information:
www.packtpub.com/less-web-development-essentials/book
Currently, he writes a blog (http://bassjobsen.weblogs.fm/), programs LBS for mobile devices (http://www.gizzing.nl), makes cool websites (such as http://www.streetart.nl/), and counsels Jamedo Websites (http://www.jamedowebsite.nl/) in setting up the technical environment and requirements for their business.

You can also check out his Bootstrap WordPress Starters Theme (JBST) and other projects at GitHub at https://github.com/bassjobsen.

"I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it."

–Bill Gates

Acknowledgments

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

Recently, I reviewed Getting Started with Zurb Foundation 4 by Andrew D. Patterson and Learning Zurb Foundation by Kevin Horek. After finishing this book, I will start writing Less Web Development Cookbook for Packt Publishing.

Although I have written many blogs and technical project requirements in the past years, this is the first book I have written to be published. Writing this book wasn't possible without the support of my family, Caroliene, and the people of Vivent. Richard Harvey was a patient and excellent motivator and critical reader. Sruthi Kutty helped me dot the i’s and cross the t’s. Finally, I will thank the reviewers of this book, Simone Deponti, Austin Pickett, and Marcus Bointon, for their critical and valuable suggestions, which make this book even better.

For More Information:
www.packtpub.com/less-web-development-essentials/book
Less Web Development Essentials

After the introduction of HTML 4.01 in 1999, the Web changed fast. Many new devices such as tablets and mobile phones saw the light of day. Mobile Internet became faster, cheaper, and more stable. The W3C started the HTML5 working group in 2007. In December 2012, W3C designated HTML5 as a candidate recommendation. HTML5 works with CSS3. Today, all major browsers (Chrome, Safari, Firefox, Opera, IE) offer HTML5 support.

The impact of CSS3 has been huge. Nowadays, CSS3 is not only used to style your HTML documents, but CSS3 also plays an important role in the responsibility of your designs. Last but not least, CSS3 extends CSS with features such as animations and transitions.

We don't need external flash components for complex animation. Take a look at http://www.hongkiat.com/blog/css3-animation-transition-demos/ or look at the funny owl in the following screenshot:

The owl in the preceding screenshot has been built with HTML5 and CSS3 alone. The live version can wink and look by pressing the buttons.

Responsive designs allow you to build one version of your website with only one code base which functions well and looks good on different devices such as mobile phones, tablets, and desktops. There won't be any technical reason to build different mobile and desktop versions, as shown in the following screenshot:

For More Information:
www.packtpub.com/less-web-development-essentials/book
With all this new stuff, the work of the CSS (or web) developer becomes more complex. A web developer needs to know about complex CSS3, the difference between browsers and devices, animations, and other style effects. Writing correct and functional CSS code will be the first thing; keeping this code readable, maintainable, and working on all major browsers will be the second thing. CSS files grow and become untidy in the development and maintenance processes. CSS doesn't have the ability to modify the existing values or reuse common styles. Also, doing math or defining variables is not possible in CSS. This is where Less comes into the frame.

**Less (Leaner CSS)** is a dynamic stylesheet language designed by Alexis Sellier. Started in 2010, it is now maintained and extended by the Less core team. Less helps you make your CSS code maintainable, reusable, and prevent code duplications.

In this book, you will learn how to write, compile, and understand Less. We will help you do faster and more cost-effective web development. You will get practical tips to integrate Less in your current and new projects. After reading this book, you will write clear and readable CSS3 with Less. Instead of spending your time on debugging your complex CSS code for a specific device or browser, you can pay more attention to your real design tasks.

Your clients will be happy with your advanced and stable designs. This will reduce the development and maintenance time and hence the cost of designing.

Less extends CSS with functions and variables. In a semantic sense, valid CSS is also valid Less. The initial versions of Less were written in Ruby; now, Less is written in JavaScript.

For More Information:
www.packtpub.com/less-web-development-essentials/book
Less is called a CSS precompiler. This means that the end product will be used for production. The end product in this case will be valid, compact, and readable CSS code. Besides, the precompiling Less code can also compile in real time. Less offers server-side and client-side options to do this. Real-time client-side compilation via LESS.js in a modern web browser makes testing easy. Server-side compilations offer opportunities to build applications with Less as well as create dynamic CSS.

Also, others know the power of Less. Projects such as Twitter's Bootstrap and Roots, a WordPress starter theme, both rely on Less. These projects build clear and extendable frameworks with Less. You can't ignore this proof. Stop writing cumbersome CSS with bugs and browser defects and learn about Less by reading this book.

Less is open source and licensed under the Apache license. At the time of writing this book, the latest version is 1.7. The source code of Less will be maintained on GitHub. Everybody will be allowed to contribute to it. You can use Less free of charge.

**What This Book Covers**

*Chapter 1, Improving Web Development with Less*, shows how CSS3 brought advanced functions such as gradients, transitions, and animations to web designers. It also explains how, on the other hand, CSS code became more complex and difficult to maintain. Less helps you make your CSS maintainable, reusable, and prevent code duplications.

*Chapter 2, Using Variables and Mixins*, explains why variables allow you to specify widely-used values in a single place and then reuse them throughout the style sheet, thus making global changes as easy as changing one line of code. Mixins allow you to embed all the properties of a class into another class by simply including the class name as one of its properties. The chapter also explains what parametric mixins are and how to use them.

*Chapter 3, Nested Rules, Operations, and Built-in Functions*, explains the use of nested rules for making inheritance clear and for making shorter style sheets. The chapter also explains how to create complex relationships between properties and how to use the built-in functions of Less.

*Chapter 4, Avoid Reinventing the Wheel*, teaches you how Less code and mixins can become complex because they handle different browsers and devices. The chapter also explains prebuilt mixins and other sources that help you (re)use them.

*Chapter 5, Integrate Less in Your Own Projects*, teaches you how to organize your files for new projects or get the projects you maintain ready for using Less.

*Chapter 6, Bootstrap 3, WordPress, and Other Applications*, explains what Bootstrap is and shows the strength of using Less with Bootstrap. The chapter also teaches you how to build web applications with Less or integrate it in your WordPress themes.

---

**For More Information:**

Improving Web Development with Less

It is impossible to imagine modern web design without CSS. With CSS3, web designers are able to rely on advanced functions such as gradients, transitions, and animations. On the other hand, CSS code becomes more complex and difficult to maintain. Less is a CSS preprocessor that extends CSS with modern programming-language concepts. Less enables you to use variables, functions, operations, and even rule or selector nesting while coding your CSS. Less helps you write CSS with the Don't Repeat Yourself (DRY) principle. The DRY principle prevents you from repeating any kind of information in your code.

This chapter will cover the following topics:

- Introduction to CSS3
- Compiling Less into CSS
- Vendor-specific rules
- CSS3 rounded corners, animations, and gradients
- Using box-sizing border-box
- Server-side compiling and using GUI

For More Information:
www.packtpub.com/less-web-development-essentials/book
Using CSS3 for styling your HTML

In web design, you will use HTML to describe the structure of your documents and CSS language to describe their presentation, including fonts, colors, and layout. The current standard HTML5 and CSS3 versions work on most modern browsers and mobile devices. CSS3 extends the old CSS with other new selectors, text effects, background gradients, and animations. The power of CSS3, the new functionalities, and high acceptance on mobile devices using HTML5 and CSS3 make them the standard for modern web design. The combination of HTML5 and CSS3 is ideal for building responsive websites because of their high acceptance on mobile phones (and other devices).

Together, HTML5 and CSS3 introduce many new features. You will be shown the ones that are the most significant when learning about their concepts within this book.

Using CSS Selectors to style your HTML

With Less (and CSS), you can style your HTML code using selectors. CSS selectors are patterns or names that identify which HTML elements of the web page should be styled. CSS selectors play an important role in writing Less code.

For `body p.article {color:red}`, the selector here is `body p.article`. Selectors don't refer exclusively to one element. They can point to more than one element and different ones can refer to the same element. For instance, a single `p` selector refers to all the `p`-elements, including the `p`-elements with a `.article` class. In the case of conflicts, cascade and specificity determine which styles should be applied. When writing Less code, we should keep the aforementioned rules in mind. Less makes it easier to write complex CSS without changing how your website looks. It doesn't introduce any limitations on your final CSS. With Less, you can edit well-structured code instead of changing the effect of the final CSS.

CSS3 introduces many new and handy selectors. One of them is `:nth-child(n)`, which makes it possible to style, for example, every fourth paragraph's `p` tag in an HTML document. Such selectors add powerful functions to CSS3. Now we are able to perform operations with CSS alone, whereas, in the past we needed JavaScript or hardcoded styles (or classes at the very least). Again, this is one of the reasons to learn Less. Powerful selectors will make CSS more important, but CSS code also becomes cumbersome and difficult to maintain. Less will prevent this problem in CSS, even making complex code flexible and easy to maintain.


For More Information:
www.packtpub.com/less-web-development-essentials/book
Specificity, Inheritance, and Cascade in CSS

In most cases, many CSS styles can be applied on the same HTML element, but only one of them will win. W3C specifications describe the rules for which CSS styles get the most precedence and will ultimately be applied. You can find these specifications in the following section.

The rules regarding the order of importance have not significantly changed with CSS3. They are briefly mentioned to help you understand some of the common pitfalls with Less/CSS and how to solve them. Sooner or later, you will be in a situation where you're trying to apply a CSS style to an element, but its effect stays invisible. You will reload, pull out your hair, and check for typos again and again, but nothing will help. This is because in most of these cases, your style will be overruled with another style that has a higher precedence.

The global rules for Cascade in CSS are as follows:

- Find all the CSS declarations that apply to the element and property in question.
- Inline styles have the highest precedence, except for !important.
  The !important statement in CSS is a keyword used to add weight to a declaration. The !important statement is added at the end of a CSS property value. After this, check who set the declaration; styles set by the author get a higher precedence than the styles defined by the user or browser (default). Default means the styles are set by the web browser, author styles are defined by CSS in the web page, and user styles are set by the user via the settings of his or her web browser. The importance of the user is higher than the default, and the code with the !important statement (see Chapter 2, Using Variables and Mixins for its meaning in Less) will always get the highest precedence. Note that browsers such as Firefox have options to disable pages in order to use other alternatives to user-defined fonts. Here, the user settings overrule the CSS of the web page. This way of overruling the page settings is not part of the CSS precedence unless they are set using !important.

- Calculate the specificity, which is discussed in the following section.
- If two or more rules have the same precedence and specificity, the one declared last wins.

As a Less/CSS designer, you will be making use of the calculated CSS specificity in most cases.
How CSS specificity works

Every CSS declaration gets a specificity, which will be calculated from the type of declaration and the selectors used in its declaration. Inline styles will always get the highest specificity and will always be applied (unless overwritten by the first two Cascade rules). In practice, you should not use inline styles in many cases as it will break the DRY principle. It will also disable you from changing your styles on a centralized location only and will prevent you from using Less for styling.

An example of an inline style declaration is shown as follows:

```html
<p style="color:#0000ff;">
</p>
```

After this, the number of IDs in the selector will be the next indicator to calculate specificity. The `#footer #leftcolumn {` selector has 2 IDs, the `#footer {` selector has 1 ID, and so on.

```
Note that in this case, an ID is a unique selector starting with #; the selector [id=] for the same HTML element counts as an attribute. This means that `div.#unique {}` has 1 ID and `div[id="unique"]` has 0 IDs and 1 attribute.
```

If the number of IDs for two declarations is equal, the number of classes, pseudo classes, and attributes of the selector will be of importance. Classes start with a dot. For example, `.row` is a class. Pseudo classes, such as :hover and :after, start with a colon, and attributes, of course, are href, alt, id, and so on.

The `#footer a.alert:hover {}` selector scores 2 (1 class and 1 pseudo class) and the `#footer div.right a.alert:hover {}` selector scores 3 (2 classes and 1 pseudo class).

If this value is equal for both declarations, we can start counting the elements and pseudo elements. The latest variable will be defined with a double colon (::). Pseudo elements allow authors to refer to otherwise inaccessible information, such as ::first-letter. The following example shows you how that works.

The `#footer div a{}` selector scores 2 (2 elements) and the `#footer div p a {}` selector scores 3 (3 elements).

You should now know what to do when your style isn't directly applied. In most cases, make your selector more specific to get your style applied. For instance, if `#header p{}` doesn't work, then you can try adding a `#header #subheader p{}` ID, a `#header p.head{}` class, and so on.

For More Information:
www.packtpub.com/less-web-development-essentials/book
When Cascade and `:important` rules do not give a conclusive answer, specificity calculation seems to be a hard and time-consuming job. Although Less won't help you here, tools such as Firebug (and other developer tools) can make the specificity visible. An example using Firebug is shown in the following screenshot, where the selector with the highest specificity is displayed at the top of the screen and the overruled styles are struck out:

```css
.inherited from ul.nav
docs.css (line 462)

.bs-sidenav {
  text-shadow: 0 1px 0 #FFFFFF;
}

.inherited from body
bootstrap.min.css (line 7)

body {
  color: #333333;
  font-family: "Helvetica Neue",Helvetica,Arial,sans-serif;
  font-size: 14px;
  line-height: 1.42857;
}
```

An example of specificity in Firebug

### Building your layouts with flexible boxes

The Flexbox Layout (also called flexible boxes) is a new feature of CSS3. It is extremely useful in creating responsive and flexible layouts. Flexbox provides the ability to dynamically change the layout for different screen resolutions. It does not use floats and contains margins that do not collapse with their content. Unfortunately, major browsers do not offer full support for Flexbox layouts at this moment. We focus on Flexbox due to its power, and as it is an important feature of CSS, we can also produce and maintain it using Less. You can access a set of Less mixins for CSS3 Flexbox at https://gist.github.com/bassjobsen/8068034. You can use these mixins to create Flexbox layouts with Less, without using duplicate code.

For More Information:
www.packtpub.com/less-web-development-essentials/book
These mixins will not be explained in great detail now, but the following example shows how Less reduces the code needed to create a flex container. Using CSS, you might use the following code:

```css
div#wrapper {
    display: -webkit-flex;
    display: -moz-flex;
    display: -ms-flexbox;
    display: -ms-flex;
    display: flex;
}
```

However, if you use Less, the same effect can be produced by inserting the following line of code:

```less
div#wrapper { .flex-display; }
```

You can use Google Chrome to test your Flexbox layouts. At the time of writing this book, Firefox and Internet Explorer IE11 also offered full or better support for Flexbox layouts. Flexboxes have been mentioned because they have the potential to play an important role in the future of web design. For now, they are beyond the scope of this book. This book will focus on creating responsive and flexible layouts with Less using CSS media queries and grids.


### Compiling Less

After delving into the theory of CSS, you can finally start using Less. As mentioned earlier, it has the same syntax as CSS. This means any CSS code is, in fact, a valid Less code too. With Less, you can produce CSS code that can be used to style your website. The process used to make CSS from Less is called compiling, where you can compile Less code via the server side or client side. The examples given in this book will make use of client-side compiling. Client side, in this context, means loading the code in a browser and compiling Less code into CSS code using resources from the local machine. Client-side compiling is used in this book because it is the easiest way to get started while being good enough for developing your Less skills.
It is important to note that the results from client-side compiling serve only for demonstration purposes. For production and especially when considering the performance of an application, it is recommended that you use server-side precompiling. Less bundles a compiler based on Node.js, and many other GUI’s are available to precompile your code. These GUI’s will be discussed towards the end of this chapter.

Getting started with Less

You can finally start using Less. The first thing you have to do is download Less from http://www.lesscss.org/. In this book, Version 1.6 of less.js will be used. After downloading it, an HTML5 document should be created. It should include less.js and your very first Less file.

Please note that you can download the examples, including a copy of less.js, from the support files for this chapter in the downloadable files for the book on www.packtpub.com.

To start with, have a look at this plain yet well-structured HTML5 file:

```html
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Example code</title>
        <meta name="description" content="Example code">
        <meta name="author" content="Bass Jobsen">
        <link rel="stylesheet/less" type="text/css" href="less/styles.less" />
        <script src="less.js" type="text/javascript"></script>
    </head>
    <body>
        <h1>Less makes me Happy!</h1>
    </body>
</html>
```

As you can see, a Less file has been added to this document using the following code:

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

For More Information:
www.packtpub.com/less-web-development-essentials/book
When rel="stylesheet/less" is used, the code will be the same as for a style sheet. After the Less file, you can call less.js using the following code:

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

In fact, that’s all that you need to get started!

To keep things clear, html5shiv (which you can access at http://code.google.com/p/html5shiv/) and Modernizr (which you can access at http://modernizr.com/) have been ignored for now. These scripts add support and detection of new CSS3 and HTML5 features for older browsers such as IE7 and IE8. It is expected that you will be using a modern browser such as Mozilla Firefox, Google Chrome, or any version of Internet Explorer beyond IE8. These will offer full support of HTML5, CSS3, and media queries, which you will need when reading this book and doing the exercises.

You already know you should only use less.js for development and testing in most cases; there can still be use cases which do justice to the client-side use of less.js in production. To support less.js for older browsers, you could try es5-shim (https://github.com/es-shims/es5-shim/).

Now, open http://localhost/index.html in your browser. You will see the Less makes me Happy! header text in its default font and color. After this, you should open less/styles.less in your favorite text editor. The syntax of Less and CSS doesn’t differ here, so you can enter the following code into this file:

```less
h1{color:red;}
```

Following this, reload your browser. You should see the header text in red.

From the preceding code, h1 is the selector that selects the HTML H1 attribute in your HTML. The color property has been set to red between the accolades. The properties will then be applied onto your selectors, just like CSS does.

It is not necessary to have a web server that is running. Navigating to index.html on your hard drive with your browser should be enough. Unfortunately, this won’t work for all browsers, so use Mozilla Firefox in order to be sure. The examples in this book use http://localhost/map/, but this can be replaced with something similar to file:///map/ or c:\map\, depending on your situation.

Using the watch function for automatic reloading

The less.js file has a watch function, which checks your files for changes and reloads your browser views when they are found. It is pretty simple to use. Execute the following steps:

1. Add !watch after the URL you want to open.
2. Add !watch after index.html and then reload the browser window.
3. So, open http://localhost/index.html!watch in your browser and start editing your Less files. Your browser will reflect your changes without having to reload.
4. Now open less/styles.less in your text editor. In this file, write #h1{color:red;} and then save the file.
5. You should now navigate to your browser, which should show Less makes me Happy! in red.
6. Rearrange your screen in order to see both the text editor and browser together in the same window.
7. Furthermore, if you change red to blue in less/styles.less, you will see that the browser tracks these changes and shows Less makes me Happy! in blue once the file is saved.

Pretty cool, isn't it?

The examples in this code use color names instead of hexadecimal values. For example, the code uses red instead of #ff0000. The basic color names are converted to their hexadecimal value by less.js and written to the CSS. In this book, named colors are always used.

Debugging your code

As we are only human, we are prone to making a mistake or a typo. It is important to be able to see what you did wrong and debug your code. If your Less file contains errors, it won't compile at all. So, one small typo breaks the complete style of the document.

For More Information:
www.packtpub.com/less-web-development-essentials/book
Debugging is also easy with less.js. To use debugging or allow less.js to display errors, you can add the following line of code to your index.html:

```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>
```

As you can see, the line with `less = { env: 'development' };` is new here. This line contains less as a JavaScript variable used by less.js. In fact, this is a global Less object used to parse some settings to less.js. The only setting that will be used in this book is `env: 'development'`. For more settings, check out the following website: http://lesscss.org/#client-side-usage-browser-options.

`env: 'development'` also prevents Less from caching. Less doesn't cache files in the browser cache. Instead, files are cached in the browser's local storage. If `env` is set to `production`, this caching could yield unexpected results as the changed and saved files are not compiled.

To try this new setting, edit `less/styles.less` again and remove an accolade to create an invalid syntax of the h1{color:red form and then save the file.

In your browser, you will see a page like the following screenshot:

```
ParseError: missing closing `}

in styles.less on line 2, column 1:
1  p{color:red;
2
```

An example of a Less parse error

Besides syntax errors, there will also be name errors that are displayed. In the case of a name error, an undeclared function or variable would have been used.

For More Information:
www.packtpub.com/less-web-development-essentials/book
It is possible to set other settings for debugging, either in the global Less object or by appending the setting to the URL. For example, you can specify the dumpLineNumbers setting by adding the following lines of code to your HTML file:

```javascript
<script type="text/javascript">less = { env:
    'development',dumpLineNumbers: "mediaQuery"
};</script>
```

Alternatively, you can add `!dumpLineNumbers:mediaQuery` to the URL. This setting enables other tools to find the line number of the error in the Less source file. Setting this option to `mediaQuery` makes error reporting available for the FireBug or Chrome development tools. Similarly, setting this to `comments` achieves the same for tools such as FireLess. For instance, using FireLess allows Firebug to display the original Less filename and the line number of CSS styles generated by Less.

FireBug, Chrome development tools, or the default browser inspect the element functions (which you can access by right-clicking on your browser screen) can also be used to see and evaluate the compiled CSS. The CSS is displayed as inline CSS wrapped inside a `<style type="text/css" id="less:book-less-styles">` tag. In the example given in the following screenshot, you will see an ID with value `less:book-less-styles`. The value of this ID have been automatically generated by Less based on the path and name of the `book/less/styles.less` Less file:

```
  <style id="less:book-less-styles" type="text/css">
    1 p {
    2     color: navy;
    3   }
  </style>
```

Less-generated CSS styles

**Example code used in this book**

In this book, you will find many code examples. Unless explicitly mentioned, the format of these examples always shows the Less code first, followed by the compiled CSS code. For instance, you can write the following lines of code in Less:

```
mixin() {
    color: green;
}
p {
    .mixin();
}
```

[19]

**For More Information:**

www.packtpub.com/less-web-development-essentials/book
This code will be compiled into the following CSS syntax:

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

**Your first layout in Less**

You must first open `first.html` (from the downloadable files for the book) in your browser and then open `less/first.less` in your text editor. In your browser, you will see a representation of a header, body, and footer.

As expected, `less/first.less` contains the Less code that will be converted into valid CSS by the `less.js` compiler. Any error in this file will stop the compiler and throw an error. Although the Less code shows some similarities to the plain CSS code, the process described here totally differs from editing your CSS directly.

The following screenshot shows you how this layout will look when opened in your web browser:
Vendor-specific rules

CSS3 introduced vendor-specific rules, which offer you the possibility of writing some additional CSS applicable for only one browser. At first sight, this seems the exact opposite of what you want. What you want is a set of standards and practicalities that work the same with every browser and a standard set of HTML and CSS which has the same effect and interpretation for every browser. These vendor-specific rules are intended to help us reach this utopia. Vendor-specific rules also provide us with early implementations of standard properties and alternative syntax. Last but not least, these rules allow browsers to implement proprietary CSS properties that would otherwise have no working standard (and may never actually become the standard).

For these reasons, vendor-specific rules play an important role in many new features of CSS3. For example, animation properties, border-radius, and box-shadow all depend on vendor-specific rules.

Vendors use the following prefixes:

- WebKit: -webkit
- Firefox: -moz
- Opera: -o
- Internet Explorer: -ms

Build rounded corners with border-radius

Border-radius is a new CSS3 property which will make many web developers happy. With border-radius, you can give HTML elements a rounded corner. In previous years, many implementations of rounded corners using images and transparency have been seen. However, these were inflexible (not fluid) and difficult to maintain.

Vendor-specific rules are required for implementation, and although rounded corners can't be handled with a single line of code, its usage definitely makes rounding corners a lot easier.

To give an element rounded corners with a radius of 10 pixels, you can use the CSS code with vendor-specific rules as follows:

- -webkit-border-radius: 10px;
- -moz-border-radius: 10px;
- border-radius: 10px;

For More Information:
www.packtpub.com/less-web-development-essentials/book
For rounded corners with different radii, use a list with values separated by spaces: 10px 5px 20px 15px. The radii are given in the following order: top-left, top-right, bottom-right, and bottom-left. By keeping these rules in mind, you will see how Less can keep your code clean.

You can open roundedcorners.html from the download section of this chapter in your browser, and open less/roundedcorners.less in your text editor. In your browser, you will see a representation of a header, body, and footer with rounded corners.

The CSS for the header in less/roundedcorners.less looks like the following code:

```css
#header{
  background-color: red;
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
}
```

You can see that using vendor-specific rules, the corners have been created with a radius of 10 pixels. If you were using CSS, you would have to repeat the vendor-specific rules three times for the header, footer, and body. In order to change these rules or add a vendor, you would also have to change the same code three times. To begin with, you will perhaps think, "Why not group the selectors?", in a fashion similar to the following code:

```css
#header, #content, #footer{
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
}
```

The preceding code is syntactically correct in order to write CSS or Less code, but as your code base grows, it won't be easy to maintain. Grouping selectors based on properties makes no sense when reading and maintaining your code. Such constructs will also introduce many duplicated and unstructured usages of the same selectors.

With Less, you are able to solve these problems efficiently. By creating a so-called mixin, you can solve the issues mentioned earlier. For the border radius, you can use the following code:

```less
.roundedcornersmixin() {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
}
```

For More Information:
www.packtpub.com/less-web-development-essentials/book
To use this mixin, you will call it as a property for the selector using the following code:

```
#header{
    background-color: red;
    .roundedcornersmixin();
}
```

The compiled CSS of this Less code will now be as follows:

```
#header{
    background-color: red;
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    border-radius: 10px;
}
```

Looking at the original code in the less/roundedcorners.less file, you can see that the preceding code wouldn't be able to work for #content. The border radius for the content is 20 pixels instead of 10 pixels, as used for the header and footer. Again, Less helps us solve this efficiently. Mixins can be called with parameters in the same way in which functions can be called in functional programming. This means that in combination with a value and a reference for this value, mixins can be called in order to set the properties. In this example, this will change to the following code:

```
.roundedcornersmixin(@radius: 10px){
    -webkit-border-radius: @radius;
    -moz-border-radius: @radius;
    border-radius: @radius;
}
```

In the .roundedcornersmixin(@radius: 10px) mixin, @radius is our parameter, and its default value will be 10px.

From this point onwards, mixins can be used in your code. The .roundedcornersmixin(50px); statement will set the corners with a radius of 50px and the .roundedcornersmixin(); statement will do the same with a radius of 10px (default).

Using this, you can rewrite less/roundedcorners.less so that it changes to the following code:

```
/* mixins */
.roundedcornersmixin(@radius: 10px){
    -webkit-border-radius: @radius;
    -moz-border-radius: @radius;
    border-radius: @radius;
}
```

For More Information:
www.packtpub.com/less-web-development-essentials/book
Improving Web Development with Less

The `less/roundedcornersmixins.less` file from the downloads section contains a copy of this code. To use this, you also have to change the reference in your HTML file to `<link rel="stylesheet/less" type="text/css" href="less/roundedcornersmixins.less" />

Note that this code leaves out the general styling of the `div` and `body` tags in the HTML. These styles are only used to make the demo look good and do not actually demonstrate Less in any useful manner.

After rewriting your Less code, reload your browser or watch it if you have applied the `#!watch` trick. You will see that the output will be exactly the same. This shows you how to get the same results with Less using a more efficiently structured code.

**Preventing cross-browser issues with CSS resets**

When talking about cascade in CSS, there will no doubt be a mention of the browser default settings getting a higher precedence than the author's preferred styling. When writing Less code, you will overwrite the browser's default styling. In other words, anything that you do not define will be assigned a default styling, which is defined by the browser. This behavior plays a major role in many cross-browser issues. To prevent these sorts of problems, you can perform a CSS reset. The most famous browser reset is Eric Meyer's CSS Reset (accessible at [http://meyerweb.com/eric/tools/css/reset/](http://meyerweb.com/eric/tools/css/reset/)).
CSS resets overwrite the default styling rules of the browser and create a starting point for styling. This starting point looks and acts the same on all (or most) browsers. In this book, normalize.css v2 is used. Normalize.css is a modern, HTML5-ready alternative to CSS resets and can be downloaded from http://necolas.github.io/normalize.css/. It lets browsers render all elements more consistently and makes them adhere to modern standards.

To use a CSS reset, you can make use of the @import directive of Less. With @import, you can include other Less files in your main Less file. The syntax is @import "{filename}". By default, the search path for the directives starts at the directory of the main file. Although setting alternative search paths is possible (by setting the path's variable of your Less environment), it will not be used in this book.

The example Less files in this book will contain @import "normalize.less"; in the first few lines of the code. Again, you should note that normalize.less does contain the CSS code. You should pay particular attention to the pros of this solution!

If you want to change or update the CSS reset, you will only have to replace one file. If you have to manage or build more than one project, which most of you should be doing, then you can simply reuse the complete reset code.

Creating background gradients

A new feature in CSS3 is the possibility of adding a gradient in the background color of an element. This acts as a replacement for complex code and image fallbacks.

It is possible to define different types of gradient and use two or more colors. In the following figure, you will see a background gradient of different colors:

![Gradient Background](A gradient example (from W3schools.com)

In the next example, you can use a linear gradient of two colors. The background gradients use vendor-specific rules.

You can make use of the example code from the rounded corners example to add gradients to it.

The first step is to copy or open less/gradient.less and add a new mixin at the start of this file as shown in the following code:

```less
/* Mixin */
.gradient (@start: black, @stop: white, @origin: left) {
  background-color: @start;
```

For More Information:

www.packtpub.com/less-web-development-essentials/book
background-image: -webkit-linear-gradient(@origin, @start, @stop);
background-image: -moz-linear-gradient(@origin, @start, @stop);
background-image: -o-linear-gradient(@origin, @start, @stop);
background-image: -ms-linear-gradient(@origin, @start, @stop);
background-image: linear-gradient(@origin, @start, @stop);
}

This will create gradients from the left (@origin) to the right with colors from @start to @stop. This mixin has default values.

IE9 (and its earlier versions) do not support gradients. A fallback can be added by adding background-color: @start;, which will create a uniform colored background for older browsers.

After adding the mixin to your code, you can call on it for our #header, #body, and #footer selectors as shown in the following code:

```
#header{
  background-color: red;
  .roundedcornersmixin();
  .gradient(red, lightred);
}
#content{
  background-color: white;
  min-height: 300px;
  .roundedcornersmixin(20px);
  .gradient();
}
#footer{
  background-color: navy;
  .roundedcornersmixin(20px);
  .gradient(navy, lightblue);
}
```

For example, if you renamed the Less file to less/gradient.less, you would have also had to change the reference in your HTML file to the following code:

```
<link rel="stylesheet/less" type="text/css"
  href="less/gradient.less" />
```
If you now load the HTML file in the browser, your results should be like the following screenshot:

Gradients in the header, content, and footer from the example code

CSS transitions, transformations, and animations

Another new feature in CSS3 is the presence of transitions, transformations, and animations. These functions can replace the animated images, flash animations, and JavaScripts in the existing or new web pages. The difference between transitions, transforms, and animations isn't trivial. Animations are constructed with a range of @keyframes, where each @keyframes handles different states of your element in time. Transitions also describe the state of element between start and end. Transitions are mostly triggered by CSS changes, such as a mouse over (hover) of an element.

For More Information:
www.packtpub.com/less-web-development-essentials/book
To make things clear, it is important to keep in mind the button that is about to be pressed. The button will have two states: pressed and not pressed. Without transitions and animations, we are enabled to style these states only. The color of the button is white, and its color becomes red when you hover the mouse over it. (In CSS terms, its state becomes hovered by adding the :hover pseudo class.) In this case, the transition describes how the hovered button becomes red. For example, the change in color from white to red in two seconds (which makes it pink halfway) shows that the start of the color change is slow and changes faster as time passes. Using animations here enables us to describe the state of the button for every time interval between the start and end. For example, you don't have to change the color from white to red, but the change covers all the states, from white, blue, green, and finally to red.

**Transformations** change the position of an element and how it looks. They do not depend on the state of the element. Some of the possible transformations are **scaling**, **translating** (moving), and **rotating**.

In practice, we use a combination of animations, transformations, and/or transitions in most situations. Also, in this case, vendor-specific rules will play an important role.

Now, a transformation will be added to our example.

Using the example code with rounded corners and gradients, copy the following code to `less/transition.less` or open `less/transition.less` and add the following code to the beginning of the file:

```less
/* Mixin */
.transition (@prop: all, @time: 1s, @ease: linear) {
-webkit-transition: @prop @time @ease;
-moz-transition: @prop @time @ease;
-o-transition: @prop @time @ease;
-ms-transition: @prop @time @ease;
transition: @prop @time @ease;
}
```

This **mixin** has three variables; the first will be the **property** (@prop) that you will change. This can be `height`, `background-color`, `visibility`, and so on. The default value `all` shouldn't be used in the production code as this will have a negative effect on performance. @time sets the duration in milliseconds or seconds with `s` appended to it. The last variable, @ease, sets the **transition-timing-function** property. This function describes the value of a property, given that a certain percentage of it has been completed. The transition-timing-function property describes the completeness of the transition as a function of time. Setting it to `linear` shows the effect with the same speed from start to end, while `ease` starts slow and ends slow, having a higher speed in the middle. The predefined functions are `ease`, `linear`, `ease-in`, `ease-out`, `ease-in-out`, `step-start`, and `step-end`.

---

For More Information:

Now, you can edit `less/transition.less` to use this mixin. You can set the background color of the body when you hover over it. Note that you don't need to use the transition to change the gradient color but rather change the `background-color` attribute. You are using `background-color` because `transition-duration` doesn't have a visible effect on the gradient. The code of the `background-color` transition is as follows:

```css
#content{
  background-color: white;
  min-height: 300px;
  .roundedcornersmixin(20px);
  .transition(background-color,5s);
}
#content:hover{
  background-color: red;
}
```

If you renamed the Less file, for example, to `less/transition.less`, you would also have to change the reference in your HTML file to the following code:

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

If you load the HTML file in the browser, you will be able to see the results in the browser. Move your mouse over the content and see it change from white to red in 5 seconds.

Finally, a second example that rotates the header can be added. In this example, you will use `@keyframes`. Using `@keyframes` will be complex. So, in this case, you can define some vendor-specific rules and add these animation properties to `#header` as follows:

```css
@-moz-keyframes spin { 100% { -moz-transform: rotate(360deg); } }
@-webkit-keyframes spin { 100% { -webkit-transform: rotate(360deg); } }
@keyframes spin { 100% { -webkit-transform: rotate(360deg);
  transform:rotate(360deg); } }
#header{
  -webkit-animation:spin 4s linear infinite;
  -moz-animation:spin 4s linear infinite;
  animation:spin 4s linear infinite;
}
```

For More Information:

www.packtpub.com/less-web-development-essentials/book
You can add the preceding code to our example files or open less/keyframes.less.

If you renamed the Less file, for example, to less/keyframes.less, you also have to change the reference in your HTML file to the following code:

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

Now, load the HTML file in the browser and watch your results. Amazing, isn't it? With a little bit of creative thinking, you will see the possibilities of creating a rotating windmill or a winking owl using only CSS3. However, the first thing that should be done is to explain the code used here in more detail. As mentioned earlier, there are many cases in which you would make combinations of animations and transformations. In this example, you also get to animate a transformation effect. To understand what is going on, the code can be split into three parts.

The first part is @keyframes, shown in the following code, which describe the value of the CSS properties (transformation in this case) as a function of the percentage of the animation completeness:

```css
@keyframes spin { 100% { -webkit-transform: rotate(360deg); transform:rotate(360deg); } }
```

These keyframes have been given the name reference spin, which is not a special effect but only a chosen name. In the preceding example, a state of 100 percent completeness is described. At this state, the animated element should have made a rotation of 360 degrees.

This rotation is the second part that needs our attention. The transformation describes the position or dimensions of an element in the space. In this example, the position is described by the number of degrees of rotation around the axis, 360 degrees at 100 percent, 180 degrees at 50 percent, 90 degrees at 25 percent, and so on.

The third part is the animation itself, described by: animation:spin 4s linear infinite;. This is the shorthand notation of settings of the subproperties of the animation property. In fact, you can write this as the following code, without the vendor-specific rules:

```css
animation-name: spin;
animation-duration: 4s;
animation-timing-function: linear;
animation-iteration-count: infinite;
```

For More Information:
www.packtpub.com/less-web-development-essentials/book
You can use these three parts to build a complete animation. After doing this, you can extend it. For example, add an extra keyframe, which makes the time curve nonlinear, as follows:

```css
@keyframes spin {
  50% { transform: rotate(10deg); }
  100% { transform: rotate(360deg); }
}
```

You can add a second property using `background-color`. Don't forget to remove the gradient to see its effect. This is shown in the following code:

```css
@-moz-keyframes spin {
  50% { transform: rotate(10deg); background-color: green; }
  100% { transform: rotate(360deg); }
}
//.gradient(red,yellow);
```

You will have noticed that the complete profit of using *Less* isn't realized here. You will have to write the `@keyframes` definition repeatedly due to its variable animation name. In *Chapter 4, Avoid Reinventing the Wheel*, a solution will be provided to you for this.

Unfortunately, browser support for transitions, transformations, and animations is not great and varies between browsers. Google Chrome does not support CSS 3D transforms, Firefox lacks support for CSS Filters, and IE9 (and earlier versions) don't support them at all. To solve this, many developers look to jQuery to support their animations. The `jQuery.animate()` function allows us to change the CSS properties of the elements using JavaScript. You can still use *Less* to set the initial CSS. An alternative for this will be to use `animate.css` (which you can access at https://github.com/daneden/animate.css); this cross-browser library of CSS animations gets converted into *Less* code with a jQuery fallback.

**Box-sizing**

The **box-sizing** property is the one that sets the CSS-box model used for calculating the dimensions of an element. In fact, box-sizing is not new in CSS, but nonetheless, switching your code to `box-sizing: border-box` will make your work a lot easier. When using the `border-box` settings, calculation of the width of an element includes border width and padding. So, changing the border of padding won't break your layouts. You can find a copy of the code used in this section in `boxsizing.html` from the download files.
Nowadays, most web designs use a grid. Grids split your design into columns of equal size. This helps you make things clear and build responsive interfaces. Depending on the available screen size (or width), you can show your content and navigation with a different representation of the same columns.

To handle different screen sizes, some parts of your website will have fluid width or height. Other elements, such as borders, gutters, and the white space, should have a fixed width. The combination of fluid widths as a percentage of the screen width (or viewport) with fixed widths becomes complex. This complexity will be due to the fact that browsers use different calculations for padding and margins of elements.

In order for you to see this, look at the following example. A container of 500 pixels width has been created. Inside this container, you can add two rows and split the second row into two parts of 50 percent (or half) width.

```
<div class="wrapper" style="width:300px;"> 
  <div style="background-color:red;width:100%;">1</div>
  <div style="background-color:green;width:50%;float:left;">2</div>
  <div style="background-color:blue;width:50%;float:right;">3</div>
</div>
```

This will now look like the following screenshot:

![An HTML wrapper](image)

The current structure doesn't show a problem until you add some padding, which is used to construct some space or a border between the two columns on the second row (numbers 2 and 3 in the image of the HTML wrapper). The padding and the border will break our layout as follows:

```
<div class="wrapper" style="width:300px;"> 
  <div style="background-color:red;width:100%;">1</div>
  <div style="background-color:green;width:50%;float:left;border:5px solid yellow;"">2</div>
  <div style="background-color:blue;width:50%;border:5px solid yellow;float:right;"">3</div>
</div>
<br>
<div class="wrapper" style="width:300px;"> 
  <div style="background-color:red;width:100%;">1</div>
</div>
```

For More Information: 
www.packtpub.com/less-web-development-essentials/book
Finally, the output of this code should look like the following screenshot:

```
<div style="background-color:green;float:left;width:50%;padding-right:5px;">
  <div style="background-color:yellow;">2</div>
</div>
<div style="background-color:blue;width:50%;padding-right:5px;float:right;">
  3
</div>
```

A broken layout due to padding and borders

A similar action can be performed, except that the wrappers can be wrapped inside an extra wrapper. The `box-sizing: border-box;` declaration can then be applied to this. Now, the results should look like the following screenshot:

```
1 2 3
```

A layout with box-sizing: border-box

As you can see, the padding and borders are subtracted by 50 percent from the parent. This will make the calculation a lot easier. Of course, you can do the calculating yourself once the parent container wrapper has a fixed width. If the parent has 300 pixels, then 50 percent of this will be 150 pixels. Taking away the padding and the width of the border will give you the fixed size of a column. This won't work when your parent has a fluid width (the percentage of the viewport). Fluid layouts change their width with the width of the screen. If your screen becomes smaller, then all the elements become smaller too and the percentage stays equal. By doing calculations for all the possible screen sizes to find the real size of a column that allows all of your elements to align, you will quickly find this to be a long, challenging, and arduous process.

For More Information:

For these reasons, you should make use of `box-sizing: border-box;` for all the examples in this book. Please note that box-sizing has to also be defined by vendor-specific rules as follows:

- `-webkit-box-sizing: border-box;`
- `-moz-box-sizing: border-box;`
- `box-sizing: border-box;`

In this example, the Less code will be as follows:

```less
// Box sizing mixin
.box-sizing(@boxmodel) {
  -webkit-box-sizing: @boxmodel;
  -moz-box-sizing: @boxmodel;
  box-sizing: @boxmodel;
}

// Reset the box-sizing
*,
*:before,
*:after {
  box-sizing(border-box);
}
```

This code has been added into a separate file called `boxsizing.less`. From now on, the basics of our Less files will now contain the following code:

```less
@import: "normalize.less";
@import: "boxsizing.less";
```

In the following chapters, you will learn more about organizing your Less code into files.

**Server-side compiling**

You have taken the first few steps towards Less development already. As explained earlier, client-side compiling has been used. However, client-side compiling with less.js shouldn't be used on real websites. This is because despite making your development easy and fast, compiling your Less files for every page request (or in fact, initial page load per user) will actually slow down your website.
For the production environment, it is required that you compile your files and serve the final CSS file to the browser. The term server side can be somewhat misleading. Server side in this context means a compiled CSS code is sent to the client's browser instead of Less code, which has to be compiled in the client's browser by less.js before it is shown. You should precompile your Less code. By copying and pasting the results of less.js to a file and including this as a CSS file in your HTML files, you should have the same effect, except that your CSS is not minimized.

Less bundles a command-line compiler. Installing and using it is simple using the following command:

```
>> npm install -g less
>> lessc styles.less styles.css
```

The package manager for the Node JavaScript platform is npm. Node enables you to run Java scripts without a browser. Node and npm run on Windows, Mac OS X, and other Unix/*nix machines. You will find the Node.js source code or a prebuilt installer for your platform by visiting http://nodejs.org/download/. To install npm, please read the instructions in the README file by visiting https://www.npmjs.org/doc/README.html.

Use the –help function to get a list of options you can use with the following command-line compiler:

```
>> lessc –help
```

lescc styles.less styles.css compiles styles.less to styles.css. The links to styles.css in your HTML after successfully compiling it are then shown as follows:

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

### Compressing and minimizing your CSS

After compilation, the CSS code is clean and readable. When taking this code into production, you have to compress and minimize it in order to increase the loading speed and save on the bandwidth as well. The basic steps for compressing and minimizing the CSS code are removing comments, white spaces, and other unnecessary code. The results won't be easy to be read by a human, but this doesn't matter because you can use the Less files to update or modify the CSS.

For More Information:  
www.packtpub.com/less-web-development-essentials/book
The Less command-line compiler has two options for compressing and minimizing. The first option (-x or --yui-compress) uses the YUI CSS Compressor (which you can access at http://yui.github.io/yuicompressor/css.html) and the second option (--clean-css) uses clean-css (which you can access at https://github.com/GoalSmashers/clean-css). You cannot use both options together. Clean-css claims to be faster, and until recently, you would not have found much difference in the compression. By compiling keyframes.less from the previous example, including normalize.less and boxsizing.less, the result will have a size of 4377 bytes. With clean-css, this drops to 3516 bytes, whilst YUI gives 3538 bytes. Since Version 1.5.0 of Less, clean-css is the compiler's default option.

Graphical user interfaces
Some of you will prefer a Graphical User Interface (GUI) instead of command-line compiling. There are many GUls available for different platforms in order to edit and compile your Less code. All of them cannot be mentioned here. Instead, the following is a list of the most positive noticeable ones:

- WinLess is a Windows GUI for less.js.
- SimpLESS is a cross-platform editor and compiler with many functions, including the automatic addition of vendor-specific rules to your code.
- CodeKIT is a GUI for Mac (OS X). It compiles many languages including Less. It includes optimizations and browser previews.
- The last one mentioned is Crunch! Crunch! is also a cross-platform compiler and editor.

When choosing a GUI for Less development, always check which version of less.js it uses. Some GUI's are built on older versions of less.js and don't support the latest features.

Web developers using Visual Studio should check out Web Essentials. Web Essentials extends Visual Studio with a lot of new features, including Less. Also, other IDEs such as PHPStorm have built-in Less compilers. There is a Less plugin for Eclipse also.
Summary

In this chapter, you refreshed and extended your knowledge about CSS3. You learned how to compile your Less code on the client side. Furthermore, you have written the code that allows you to have rounded corners, gradients, and animations in Less, so you can now witness the profits of using Less and take the crucial initial steps to organize and plan your new projects. You witnessed why you would want to use CSS resets, how to compile these into Less code, as well as how the box-sizing border-box can make your job easier. You also saw what a mixin is, how to use it, and how you can import a Less file with the @import directive. Last but not least, you have learned what server-side compiling is and how to use GUIs.

In the next chapter, you will learn how to use variables in Less and how to build and use complex mixins.

For More Information:
www.packtpub.com/less-web-development-essentials/book
Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

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