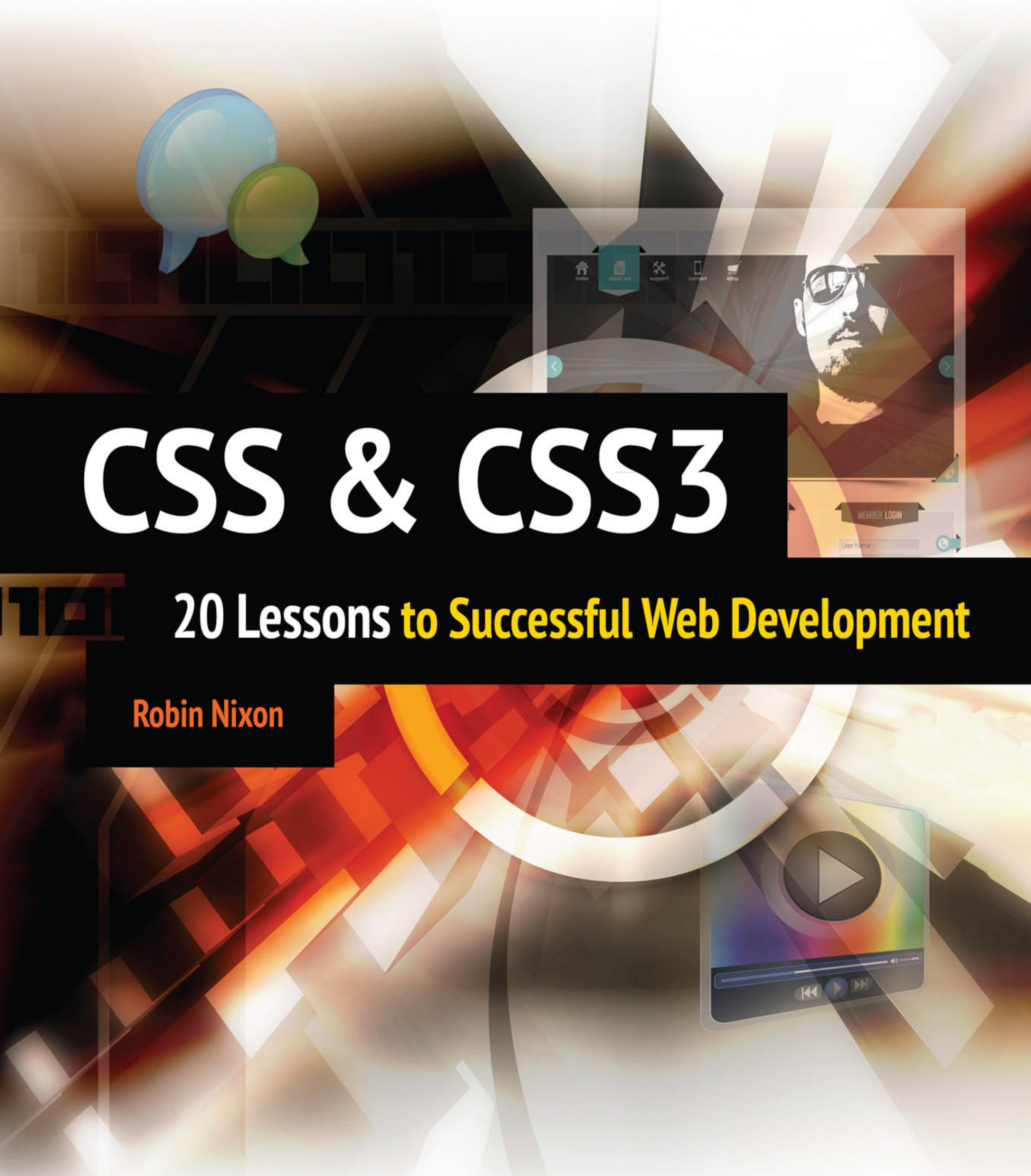
CSS & CSS3

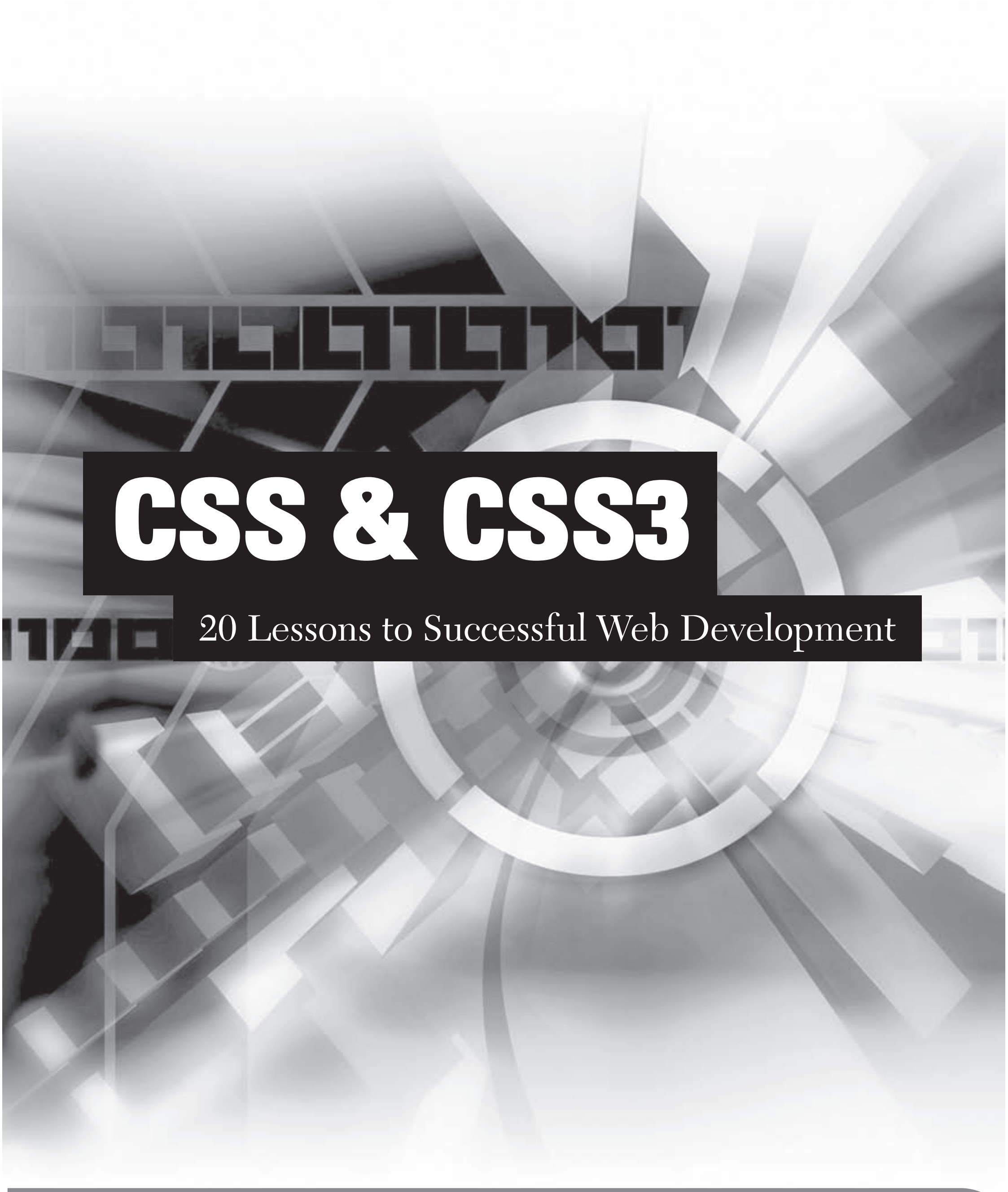
20 Lessons to Successful Web Development

Robin Nixon



Mc Graw Hill Education

Features online video tutorials for each lesson!



CSS & CSS3

20 Lessons to Successful Web Development

About the Author

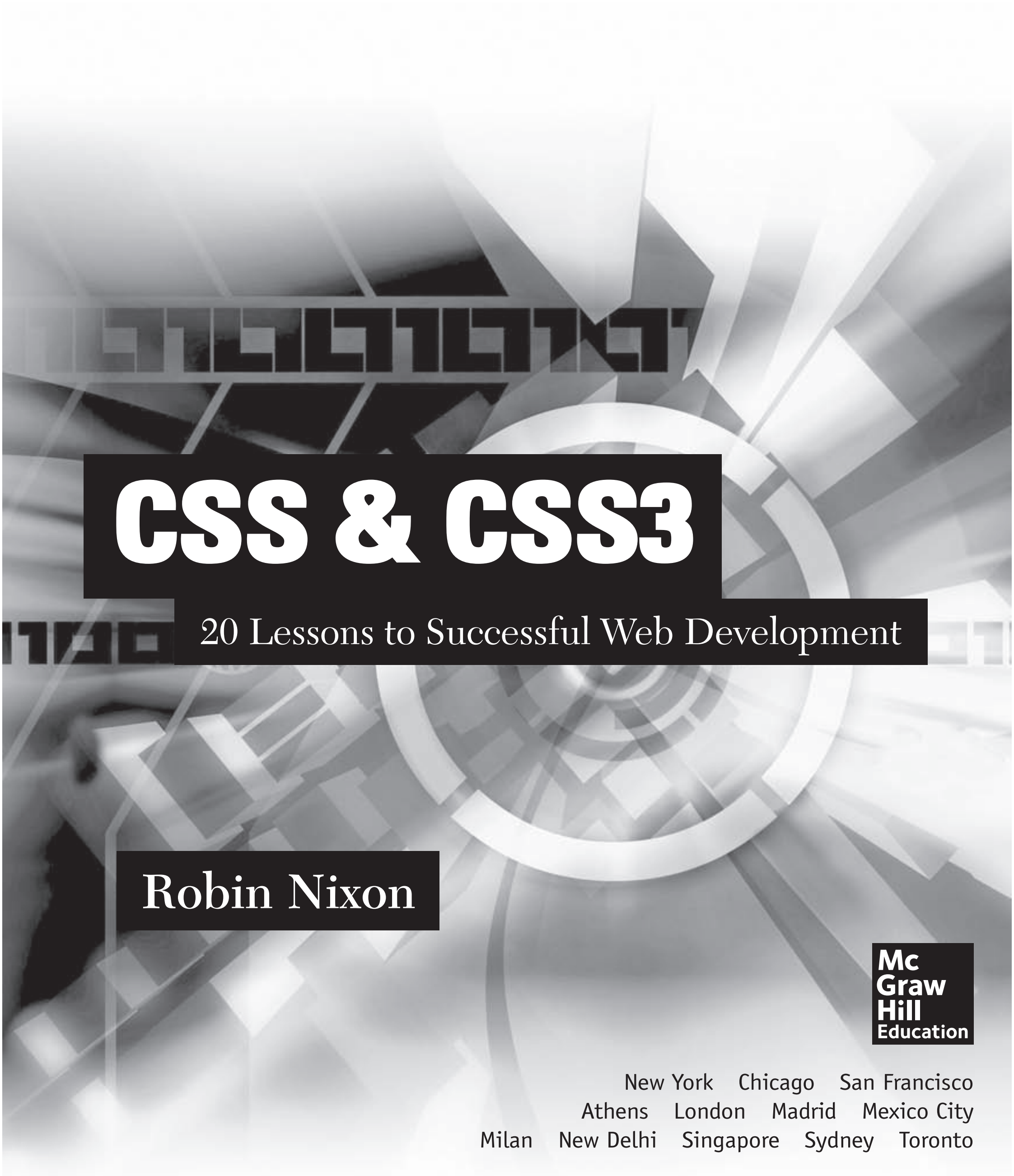
Robin Nixon is a prolific author on programming and web development (as well as psychology and motivation), with his books having been translated into numerous foreign languages—frequently topping the US and international computer book charts. He has worked with computers and technology for all his life, and began writing on the subject about 35 years ago.

He has authored hundreds of articles, and over two dozen books, and is a popular video and online instructor, with thousands of students taking his courses. Robin is also an accomplished programmer, developer, and entrepreneur, with several successful Internet startups to his name, from which he has learned a wealth of programming hints and tips, which he enjoys passing on in his expanding range of web development books, including the following titles:

- *HTML5: 20 Lessons to Successful Web Development* (McGraw-Hill Education, 2015)
- *JavaScript: 20 Lessons to Successful Web Development* (McGraw-Hill Education, 2015)
- *PHP: 20 Lessons to Successful Web Development* (McGraw-Hill Education, 2015)
- *Learning PHP, MySQL, JavaScript, CSS & HTML5* (O'Reilly, 2014)
- *Web Developer's Cookbook* (McGraw-Hill Education, 2012)
- *HTML5 for iOS and Android* (McGraw-Hill Education, 2010)

About the Technical Editor

Albert Wiersch has been writing software since the Commodore VIC-20 and Commodore 64 days in the early 1980s. He holds a Bachelor of Science degree in Computer Science Engineering and an MBA from the University of Texas at Arlington. Albert currently develops and sells software that helps web developers, educators, students, businesses, and government agencies check their HTML & CSS documents and their websites for quality problems, including many SEO (search engine optimization), mobility, and accessibility issues, with discounts made available to students. His website is at *HTMLValidator.com*.



CSS & CSS3

20 Lessons to Successful Web Development

Robin Nixon

Mc
Graw
Hill
Education

New York Chicago San Francisco
Athens London Madrid Mexico City
Milan New Delhi Singapore Sydney Toronto

ISBN: 978-0-07-185021-6

MHID: 0-07-185021-X

The material in this eBook also appears in the print version of this title: ISBN: 978-0-07-184996-8,
MHID: 0-07-184996-3.

eBook conversion by codeMantra
Version 1.0

McGraw-Hill Education eBooks are available at special quantity discounts to use as premiums and sales promotions or for use in corporate training programs. To contact a representative, please visit the Contact Us page at www.mhprofessional.com.

Information has been obtained by McGraw-Hill Education from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, McGraw-Hill Education, or others, McGraw-Hill Education does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from the use of such information.

To Julie

This page intentionally left blank

Contents at a Glance

PART I **Basic CSS**

PART II **CSS3**

This page intentionally left blank

Contents

PART II CSS3

Acknowledgments

Once again I would like to thank the amazing team at McGraw-Hill Education, with whom it is always a real pleasure to work on new book projects. In particular, I would like to thank my Sponsoring Editor Brandi Shailer, Amanda Russell for overseeing the project's development, Editorial Supervisor Jody McKenzie, Production Supervisor Jean Bodeaux, Copy Editor Megha Saini, and Jeff Weeks for the excellent cover design. Thanks also goes again to Albert Wiersch (whom I have had the pleasure of working with on a number of occasions) for his meticulous eye for detail during technical review.

This page intentionally left blank

Introduction

Why This Book?

The concept for this book grew out of Robin's extremely popular online courses in which thousands of students are enrolled. From their feedback it became evident that the reason for this popularity was that students love the way the material is broken down into easy-to-digest lessons, each of which can be completed in an hour or less. They also like the no-nonsense, jargon-free, and friendly writing style.

Now, working together, Robin and McGraw-Hill Education have further revised, updated, and developed his CSS & CSS3 course into this book, which not only will teach you everything you need to learn in 20 lessons (of less than an hour each), it also includes an average 15-minute detailed video walk-through for each lesson—comprising 5 hours of footage in total.

Watch the videos after reading the lesson to reinforce key concepts, or use the video as a primer to working through each print lesson. Used together, these course materials make learning CSS & CSS3 easier than it has ever been, and is the ideal way for you to add these essential skills to your web development toolkit.

▶ To view the accompanying video for this lesson, please visit mhprofessional.com/nixoncss/.

Who Should Read This Book?

Each chapter is laid out in a straightforward and logical manner as a lesson, with plenty of examples written using simple and clear CSS. Before moving on to each subsequent lesson, you have the opportunity to test your new knowledge with a set of 10 questions about what you have just learned. You can also work along with every lesson by watching its accompanying video tutorial.

Even if you have never used any CSS before, you will still learn everything you need from this book, because the first section provides a thorough grounding in what CSS is, what it can do for you, and how to use it. And if you have used CSS before, it will act as a great revision source to ensure you are fully up-to-date before moving on to learning all the latest additions to CSS3.

Between the lessons, the self-test questions, and the videos, this course will ensure that you become expert at CSS very quickly.

To save you typing them in, all the example files from the book are saved in a freely downloadable zip file available at the companion website: *20lessons.com*.

What This Book Covers?

This book covers every aspect of CSS, starting with basic syntax and language rules, such as where and how you include CSS in your web documents. Then the differences between styling elements by type, class, and ID are explained, along with how to refine the elements to which CSS will be supplied using selectors.

The term cascade (in cascading style sheets) is fully explained, as is how to create professional results with fonts and typography, and how to lay out compelling pages. All the CSS and CSS3 selectors are detailed, along with the new CSS3 ways to manage colors, backgrounds, borders, opacity, and more. How to transform and animate elements in 2D and 3D is also explained, with simple examples that you can easily incorporate in your own websites.

By the time you finish the book's 20 lessons, you'll have a thorough grounding in CSS, and be able to use it to ensure your web pages look as good as possible.

How to Use This Book?

This book has been written in a logical order so that each lesson builds on information learned in the previous ones. You should begin at Lesson 1 and then work sequentially through the book, proceeding to the next lesson only when you can correctly answer the self-test questions in the previous one.

If you already use CSS, you may wish to just browse through the first few lessons before tucking into the CSS3 section.

Lessons should take you less than an hour to finish, including viewing the accompanying video walk-through provided with each one. With over 5 hours of video in total, that's an average of 15 minutes dedicated to each lesson.

How Is This Book Organized?

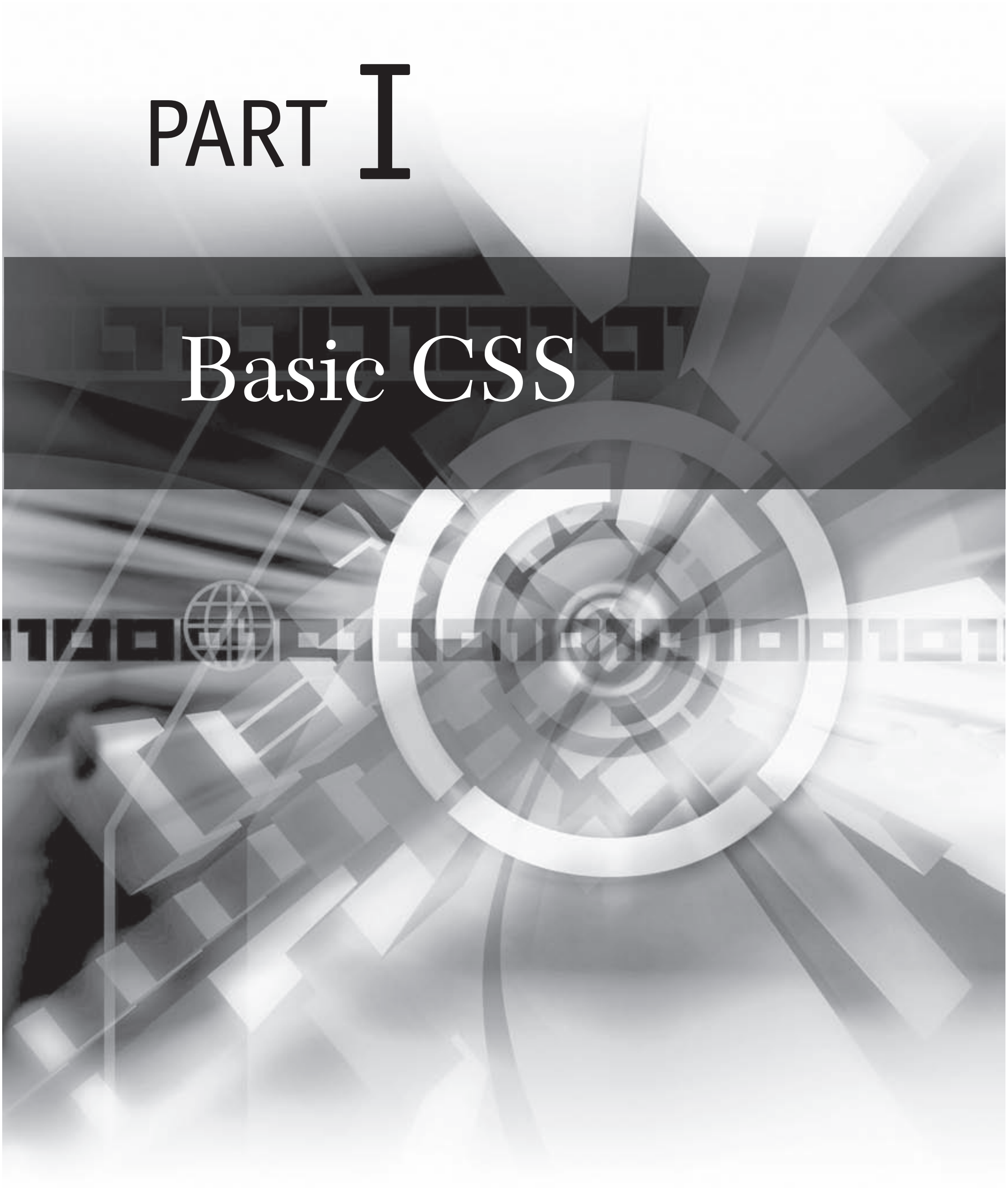
The first section of this book covers everything you need to know as a newcomer to CSS, including these lessons: Introduction to CSS, Learning the CSS Rules, Applying Declarations to IDs and Classes, Accessing Selectors, Working with the Cascade, Selecting Fonts and Typography, Manipulating Color and Position, Handling Pseudo-Selectors and Using Shorthand Properties, and Understanding the Box Model.

The second section moves on to explaining everything that's new in CSS3, including these lessons: Introduction to CSS3, Using Selectors and Attribute Selectors, Setting Backgrounds, Attaching Borders, Controlling Box Shadows, Overflow, and Columns, Adding Colors and Opacity, Creating Text Effects and Changing the Box Model, Linking to Web Fonts, Making 2D Transformations, Applying Specific Transformations, and Directing 3D Transformations. The appendix lists all the answers to the self-test questions in each chapter.

This page intentionally left blank

PART I

Basic CSS



This page intentionally left blank

LESSON 1

Introduction to CSS

To view the accompanying video for this lesson, please visit mhprofessional.com/nixoncss/.

When CSS was invented it was based around a Document Object Model (DOM), a means of separating out all the different elements within a web page into discrete objects, each with its own properties and values. This led logically to the introduction of style sheets, enabling the content of a web page to be completely separated from its styling; it also made HTML documents easily modifiable by languages such as JavaScript to provide dynamic user interaction.

Because web pages use a DOM, it is easy for you to style every aspect of it using CSS. For example, each heading will be within pairs of tags such as `<h1>` and `</h1>`, and a single CSS declaration can set the styling of all such occurrences within a document, changing the font used, its size, any font decoration, and so on.

This lets you completely change the design of a page without altering the HTML. Some style settings can even apply dynamic effects to page elements, such as changing their color and other properties when the mouse passes over them, or create transition effects by using proprietary browser extensions.

Note The example files from this book are in a file you can download at *20lessons.com*. The files for this lesson are saved in it as *embeddedstyles.htm*, *example.htm*, *importedstyles.htm*, *importedstyles2.htm*, *styles.css*, *styletag.htm*, *usingclasses.htm*, and *usingids.htm*.

How the Document Object Model Works

The DOM separates different parts of an HTML document into a hierarchy of objects, each one having its own properties. The term *property* is used for referring to an attribute of an object, such as the HTML it contains, its width and height, and so on.

The outermost object possible is the window object, which is the current browser window, tab, iframe, or popped-up window. Underneath this is the document object, of which there can be more than one (such as several documents loaded into different iframes within a page). And inside a document there are other objects such as the head and body of a page.

Within the head there can be other objects such as the title and meta objects, while the body object can contain numerous other objects, including headings, anchors, forms, and so forth. For example, Figure 1-1 shows a representation of the DOM of an example document, with the document shown inside the outer window, and having the title `Hello`, a `<meta>` tag in the head, and three HTML elements (a link, a form, and an image) in the body section.

Of course, even the simplest of web pages has more structure than is shown here, but it serves to illustrate how the DOM works; starting from the very outside is the window, inside which there's a single document, and within the document are the various elements or objects, which connect to each other.

In the figure, values are shown with a darker background and in italics. For example, the value `robots` is the value of the property `name`, which is a property of `<meta>`, and so on. Although it isn't shown in the figure, the `<meta>` tag should have another matching property called `content`, which would contain a string specifying which robots (web crawlers) may access the web page.

Other properties are `href,` which has a value of *`http://google.com`* (and is itself a property of `<meta>`, and so on), and `<title>,` which has the value `Hello`. All the other items are objects or object argument names. If the figure were to extend farther down and sideways, other objects and properties attached to the ones shown would come into view. A couple of the places where these would appear are shown by unconnected dotted lines.

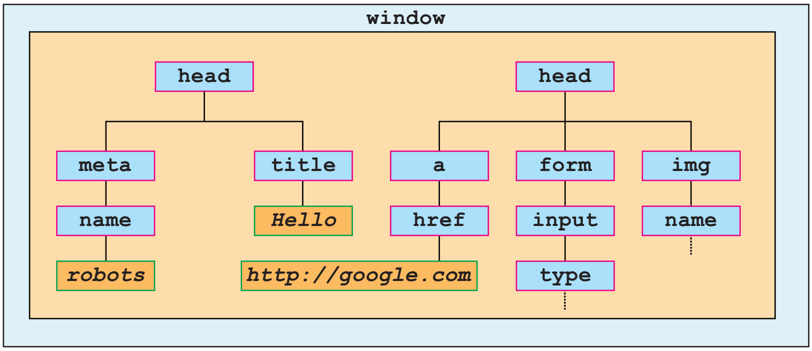


FIGURE 1-1 Example of a DOM showing head and body sections

Note In HTML tags you can generally use the single or double quotation marks interchangeably. Therefore `<a href="http://google.com">` is equivalent to `<a href='http://google.com'>`. Wherever possible, though, I tend to use single quotes because they don't require pressing the Shift key to type them in. Also there are sometimes occasions when you need two levels of nested quotes, where I would then choose double quotation marks for the outer string, and then apply single quotes within it, like this: `<p style="font-family:'Times New Roman';">`.

Representing this as HTML, the structure of the head section looks like this:

```
<head>
  <meta name='robots' content='index, follow'>
  <title>Hello</title>
</head>
```

And the body section of HTML might look like this:

```
<body>
  <img src='/images/welcome.jpg'>
  <a href='http://google.com'>Visit Google</a>
  or enter your username and password to continue...
  <form id='login' method='post' action='login.php'>
    <input type='text' name='name'>
    <input type='password' name='password'>
    <input type='submit'>
  </form>
</body>
```

Remembering that these two sections of HTML are part of the same document, we would bring them both together inside an `<html>` tag (preceded by a `!DOCTYPE` declaration), like this:

```
<!DOCTYPE html>
<html>
  <head>
    <meta name='robots' content='index, follow'>
    <title>Hello</title>
  </head>
  <body>
    <img src='/images/welcome.jpg'>
    <a href=http://google.com>Visit Google</a>,
    or enter your username and password to continue...
    <form id='login' method='post' action='login.php'>
      <input type='text' name='name'>
      <input type='password' name='password'>
      <input type='submit'>
    </form>
  </body>
</html>
```

Of course, all web pages are different, but they will usually follow this same form.

Correct HTML Structure and Nesting

To follow recommended HTML structure and to ensure your documents are readable by the maximum number of browsers and other clients, attribute values within tags should be contained in either single or double quotation marks like this: `<a href='http://yahoo.com'>`, even though nearly all browsers allow you to omit them, like this: `<a href=http://yahoo.com>`.

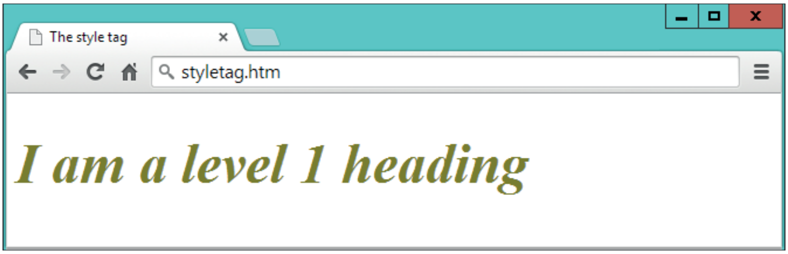
You should also close (end) every tag, and do so in the correct order. For example, you shouldn't close a document by issuing `</html>` followed by `</body>` because the proper nesting of tags would be broken by this reversal. The correct way to close a document is with `</body>`, followed by `</html>`.

About Cascading Style Sheets

Using CSS you can apply styles to your web pages to make them look exactly how you want. This works because CSS is connected to the DOM so that you can quickly and easily restyle any element. For example, if you don't like the default look of the `<h1>`, `<h2>`, and other heading tags, you can assign new styles to override the default settings for the font family and size used, or whether bold or italics should be set, and many more properties too.

One way you can add styling to a web page is by inserting the required CSS into the head of a web page between the `<head>` and `</head>` tags. So, to change the style of the `<h1>` tag you might use the following CSS:

```
<style>
  h1 {
    color      :olive;
    font-size  :18pt;
    font-family:'Times New Roman';
  }
</style>
```



I am a level 1 heading

FIGURE 1-2 A simple document with a CSS-styled heading

Within an HTML page, this might look like the following (see Figure 1-2):

```
<!DOCTYPE html>
<html>
  <head>
    <meta name='robots' content='index, follow'>
    <title>The style tag</title>
    <style>
      h1 {
        color      :olive;
        font-size  :36pt;
        font-family:'Times New Roman';
        font-style :italic;
      }
    </style>
  </head>
  <body>
    <h1>I am a level 1 heading</h1>
  </body>
</html>
```

Importing a Style Sheet

When you wish to style a whole site, rather than a single page, a better way to manage style sheets is to completely remove them from your web pages to separate files, and then import the ones you need. This lets you use different style sheets for different layouts (such as web and print) without changing the HTML.

There are a couple of different ways this can be achieved, the first of which is by using the CSS `@import` directive like this:

```
<style>
  @import url('styles.css');
</style>
```

This statement tells the browser to fetch a style sheet with the name *styles.css*. The `@import` command is quite flexible in that you can create style sheets that themselves pull in other style sheets, and so on. Just make sure that there are no `<style>` or `</style>` tags in any of your external style sheets or they will not work. External style sheets must contain only CSS and never any HTML tags.

Importing CSS from Within HTML

You can also include a style sheet with the HTML `<link>` tag like this:

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

This has the exact same effect as the @import directive, except that <link> is an HTML-only tag and is not a valid style directive, so it cannot be used from within one style sheet to pull in another. Also, it cannot be placed within a pair of <style> and </style> tags.

Just as you can use multiple @import directives within your CSS to include multiple external style sheets, you can also use as many <link> tags as you like in your HTML.

Embedded Style Settings

There's also nothing stopping you from individually setting or overriding certain styles for the current page on a case by case basis by inserting style attributes directly within HTML, like this (which results in italic blue text within the tags):

```
<h1 style='font-style:italic; color:blue;'>Hello there</h1>
```

But this should be reserved only for the most exceptional circumstances as it breaks the separation of content and presentation.

Using IDs

A better solution for setting the style of an element is to assign an ID to it in the HTML, like this:

```
<h1 id='highlight'>Hello there</h1>
```

What this does is state that the contents of the <h1> element with the ID highlight should have the style applied to it that is defined in the following rule:

```
#highlight {
  text-decoration:underline;
  color          :orange;
}
```

Note

Note the use of the # symbol, which specifies that only the element that has an ID of highlight should be styled with this statement.

Using Classes

If you would like to apply the same style to many elements, you don't have to give each one a different ID because you can specify a class to manage them all, like this:

```
<h1 class='highlight'>Hello</h1>
```

What this does is state that the contents of this element (and any other element that uses the same class) should have the style defined for the `highlight` class applied to it. Once a class is assigned you can use the following rule, either in the document's head section or within an external style sheet for setting the styles for the class:

```
.highlight {
  text-decoration:underline;
  color          :orange;
}
```

Instead of using a # symbol, which is reserved for IDs, class selectors are prefaced with a `.` (period) symbol.

Note You may wonder whether the semicolons used in CSS are actually necessary or if are they optional, as with JavaScript. The answer is yes and no. In order to separate multiple CSS statements on the same line, you must place semicolons between them. But if there is only one statement in a set of rules (or in an inline style attribute within an HTML tag), then you can omit the semicolon, as you can for the final statement in a group. However, to avoid hard-to-find CSS errors, you may prefer to always use a semicolon after every CSS declaration so that you can copy and paste them and otherwise modify properties without worrying about removing semicolons where they aren't strictly necessary or having to add them where they are required.

Summary

Now that you have learned some of the basics of CSS, in Lesson 2, I'll show you how CSS rules work, and how you can apply them in different ways.

Self-Test Questions

Using these questions, test how much you have learned in this lesson. If you don't know an answer, go back and reread the relevant section until your knowledge is complete. You can find the answers in the appendix.

1. What model is HTML based on?
2. What is the main purpose of style sheets?
3. What is the outermost object in the DOM?
4. What are the two main sections of an HTML document?
5. What types of elements are typically used in the head of an HTML document?