Getting to Know HTML and CSS

This chapter covers

- Viewing the fundamentals of HTML and CSS
- Introducing the Web Design Playground
- Learning how to construct HTML tags and CSS properties

When a jazz musician creates an improvisation, no matter how intricate, she plays by using combinations of seven musical notes (A through G). When an artist creates a picture, no matter how detailed, he paints by using combinations of three primary colors (red, yellow, and blue). When poets create verse, no matter how inventive, they write by using words that are combinations of the 26 letters of the alphabet. These examples show that creativity and play don't require elaborate resources or complex raw materials. Imagination and curiosity combined with a few building blocks are all you need to express yourself in almost any art, including the art of web page design. As you learn in this chapter and throughout this book, HTML and CSS provide those building blocks. And although there are more of those blocks than there are musical notes, primary colors, or even letters of the alphabet, there aren't too many, but more than enough to let you express yourself on an exciting modern canvas: the web.

What Is HTML?

The hardest thing about HTML by far is its name. *HTML* stands for *Hypertext Markup Language*, which sounds about as inviting as a tax audit. But it becomes a lot less intimidating when you break down its terms.

I'll begin with *hypertext*. A *link*, as I'm sure you know, is a special word or phrase (or even an image) in a web page that "points" to another web page. When you click one of these links, your browser transports you to the other page immediately. The folks who invented the web used the geeky term *hypertext link* for this special text. (The prefix *hyper* means *beyond*.) Because these hypertext links are the distinguishing features of the web, pages are often known as hypertext documents. So *HTML* has *hypertext* in it because you use it to create these hypertext documents. (It would be just as accurate to call this language WPML, or Web Page Markup Language.)

My dictionary defines *markup* as (among other things) "detailed stylistic instructions written on a manuscript that is to be typeset." For the purposes of this chapter, I can rephrase this definition as follows: "detailed stylistic instructions typed in a text document that is to be published on the World Wide Web." That's HTML in a nutshell. It has a few simple alphabetic codes—called *tags*—for detailing things such as herding text into paragraphs, creating bulleted lists, inserting images, and (of course) defining links. You type these tags in the appropriate places in an ordinary text document, and the web browser handles the dirty work of translating—or *rendering*—the tags. The result? Your page is displayed the way you want automatically.

The word *language* may be the most intimidating because it seems to imply that HTML is a programming language. Fortunately, you can rest assured that HTML has nothing to do with computer programming. Rather, HTML is a "language" in the sense that it has a small collection of words that you use to specify how you want your text to appear—as a heading or as a numbered list, for example.

In short, playing with HTML means inserting a few codes strategically between stretches of regular text in such a way that you end up with an honest-to-goodness web page. As far-fetched as this may sound to you now, you'll create a working web page by the end of this chapter, and by the end of this book, you'll have created several impressive HTML projects.

What Can You Do with HTML?

When you add HTML to a document, you're essentially giving the web browser a series of instructions that specify how you want the page to be laid out within the browser window. You use HTML to specify, in its succinct way, the overall structure of the page and to let the browser know what you want each part of the page to be. You use HTML to supply instructions similar to the following:

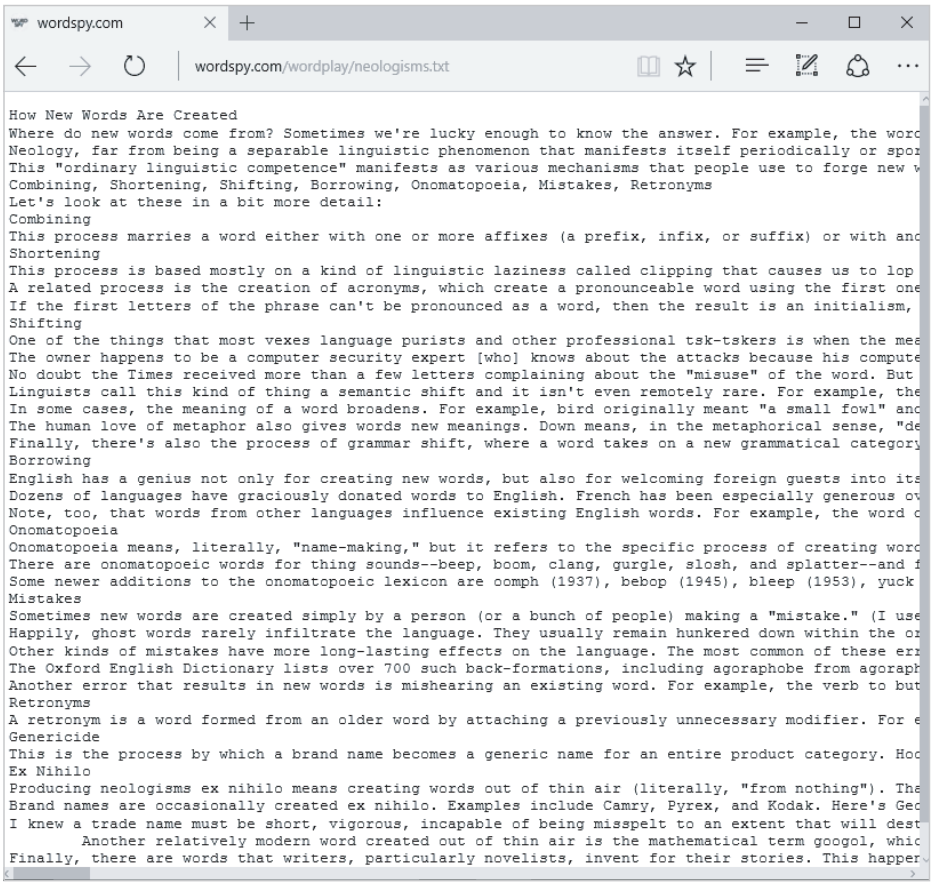
- Use this line as the main heading of the page.
- Treat these lines as subheadings.

- Make this chunk of text a separate paragraph.
- Turn these five consecutive items into a bulleted list.
- Convert these six consecutive steps to a numbered list.
- Make this phrase a link.

These instructions likely seem a bit abstract to you now, so I'll show you a concrete example of HTML in action.

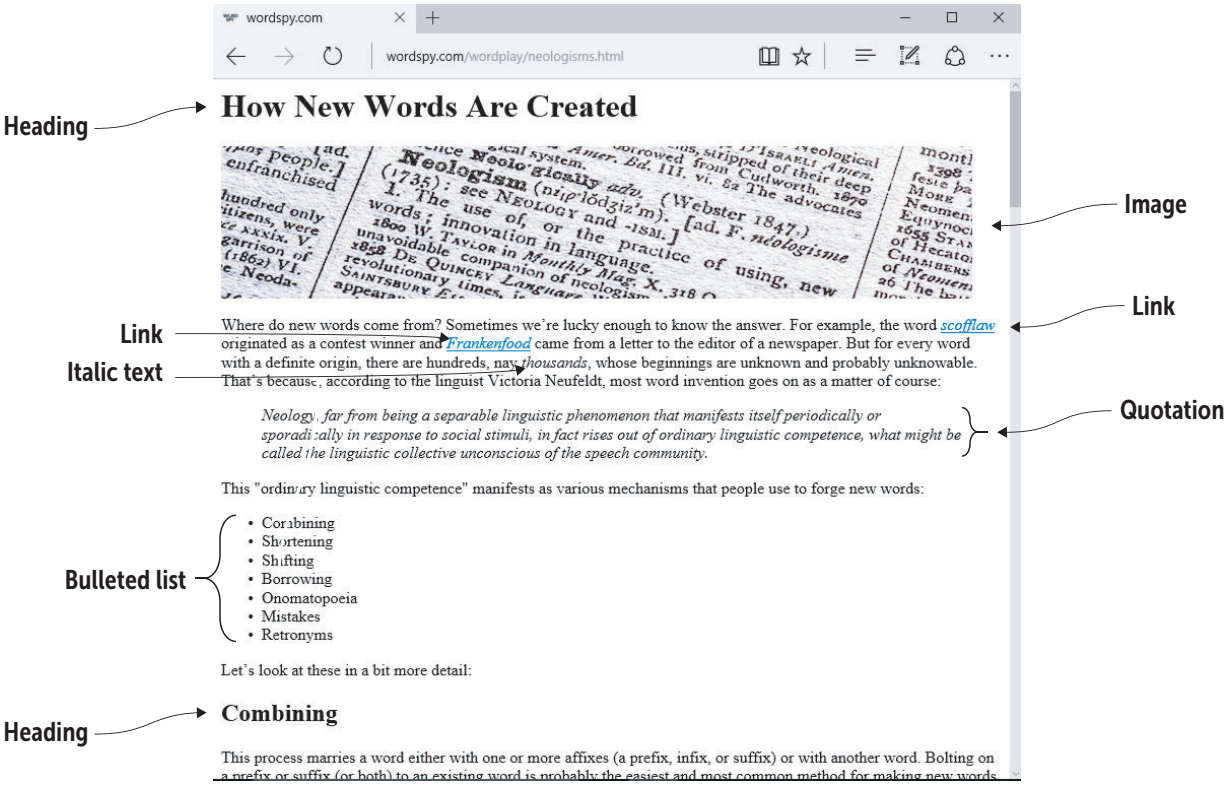
From Plain Text to HTML: An Example

Figure 1.1 shows a plain-text document displayed in a web browser. As you can see, except for the occasional line break, the browser displays a wall of unformatted, unwrapped text. This text is extremely difficult to read, and it's exceptionally hard to extract meaning from the text because it's almost entirely undifferentiated.



▶Figure 1.1
The browser can display plain-text files, but they're awfully hard to read.

HTML rides to the rescue, not only providing the means to make plain text more readable, but also allowing you to display the text in a way that your readers will find meaningful. Figure 1.2 shows the text from Figure 1.1 with some HTML applied.



▶ **Figure 1.2** With some HTML applied, the text from Figure 1.1 becomes easier to read, navigate, and understand.

Here, I've used headings to display both the article title at the top and a section title near the bottom. Notice that the section title is rendered in a type size that's slightly smaller than the main title, making the article hierarchy immediately clear. I also used HTML to add an image for visual interest. To help put the *H* in this page's HTML, I set up two of the words as links to (in this case) other sites. Although you see a bit later in this chapter that text formatting usually is the domain of CSS, you can also use HTML to add a bit of formatting flourish to your pages, such as the italics I added here. I also set up a quotation, which the browser renders indented from the regular text, and I added italics to that quotation for added differentiation. Finally, I used HTML to set up a bulleted list.

Now that you know what HTML can do, it's time to take a closer look at how you tell the browser what you want your page to look like.

What Is CSS?

When you build a house, one of the early jobs is framing, which involves putting up the basic structure for the floors, walls, and roof. That foundational framing is what you're doing when you add HTML to your page: You specify what you want to appear on the page, what you want the page's various items to be (such as a heading, paragraph, or list), and the order in which you want these items to appear.

But as a house isn't a home without finishing touches such as molding, paint, and flooring, your document isn't a modern example of a web page until you've used CSS to add some finishing work. *CSS* stands for *Cascading Style Sheets*, and as is the case with HTML, its name is more complicated than what it does. I'll break down the words, although in this case, I'll address them slightly out of order for simplicity's sake.

First, a *style* is an instruction to the browser to modify how it displays something on the page. (That something could be a word, a paragraph, or every instance of a particular HTML element.) These modifications usually are formatting-related, such as changing the typeface or the text color, but you can also use styles to control page layout and even to create animated effects. If you've ever used styles in a word processing program, you already have a good idea of what web page styles can do.

Okay, so what's a *sheet*? In the early days of publishing, firms maintained manuals that defined their preferred formatting for typefaces, headings, pulled quotes, and so on. This formatting was known as *house styles*, and the manual was called a *style sheet*. In web design, a style sheet performs essentially the same duties. It's a collection of styles that get applied to a particular web page.

To understand the *cascading* part of CSS, you need to know that, in the same way that water running down a hill can take different routes to the bottom, styles can take different routes before they get applied to an element. Some styles come from the web browser; some styles come from the user (if the user configures her browser to use a different default type size, for example); and some styles come from your style sheets. When these styles overlap, the web browser uses a complex algorithm to decide which style gets applied, and that algorithm is called the *cascade*.

You use CSS, in other words, to define how your page looks. It may seem that you use CSS only to add "eye candy" to a page, and it's certainly true that CSS offers you the tools to make only trivial or frivolous modifications. *How* your page looks, however, is every bit as important as *what* your page contains, because few people will bother to read text that's formatted poorly or incoherently.

BEWARE

The idea of the cascade is by far the most complex and convoluted aspect of CSS. I get into it later in the book (see Chapter 19), but for now, I highly recommend that you transfer it to a mental back burner until you get that far.

A Note about the Separation of Structure and Presentation

While you're trying to wrap your head around the differences between HTML and CSS, let me offer a key distinction. Although I'm generalizing somewhat, here's the basic difference between the two:

- HTML defines the overall structure of the web page.
- CSS defines the visual presentation of the web page.

Some overlap exists here (HTML can affect the presentation of the page, for example, and CSS can affect the layout), but for the most part, HTML and CSS enable you to separate structure and presentation, respectively. This distinction is important because when you keep these two aspects of a web page separate, your page will be easier to build, easier to maintain, and easier to customize.

What Can You Do with CSS?

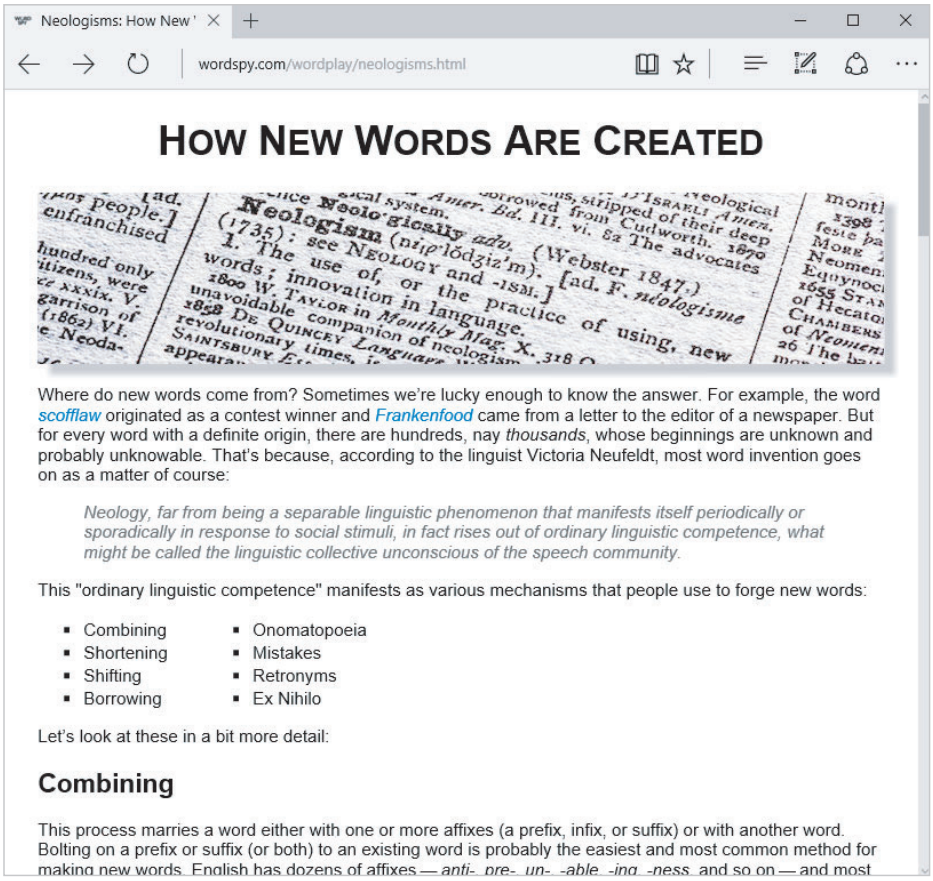
When you add CSS to a document, you're telling the web browser how you want specific elements to look. Each style is a kind of formatting instruction to the browser. You can use these instructions in a wide variety of ways that are similar to the following examples:

- Display all the links in red text.
- Use a specific font for all the headings.
- Create a bit of extra space around this paragraph.
- Add a shadow to this photo.
- Use lowercase Roman numerals for all numbered lists.
- Always display this section of text on the far-right side of the window.
- Rotate this drawing by 45 degrees.

I'll make this list more concrete by showing you an example.

From Structure to Presentation: A CSS Example

Earlier in this chapter, I took a plain-text document (Figure 1.1) and applied a bit of HTML to give it some structure and improve its readability (Figure 1.2). In Figure 1.3, I've applied a few styles to make the page look a bit nicer.



▶Figure 1.3
The example web page with a few styles applied

Here's a summary of the major styles changes I made:

- Displayed the title in a larger text size, centered and in small caps
- Added a shadow to the photo
- Made all the text slightly smaller
- Removed the underline from the links
- Displayed the quotation in lighter-color text
- Converted the bullets to a two-column list
- Increased the side margins

What Can't You Do with HTML and CSS?

Earlier, I mentioned that HTML isn't a programming language, so it's fairly straightforward to learn and to deploy it in your web pages, which is good news. The bad news is that HTML can't handle many higher-level operations *because* it's not a programming language. The list of what you can't do with HTML alone is quite long, but I'll mention the following because one or more of them may be on your to-do list:

- Get data from a server database or other remote address
- Process data submitted through a form
- Handle user accounts, logins, and passwords
- Add, hide, or remove web page elements on-the-fly

Performing tasks like these requires a programming language such as JavaScript or PHP, which are well beyond the scope of this book.

How HTML and CSS Create the Web

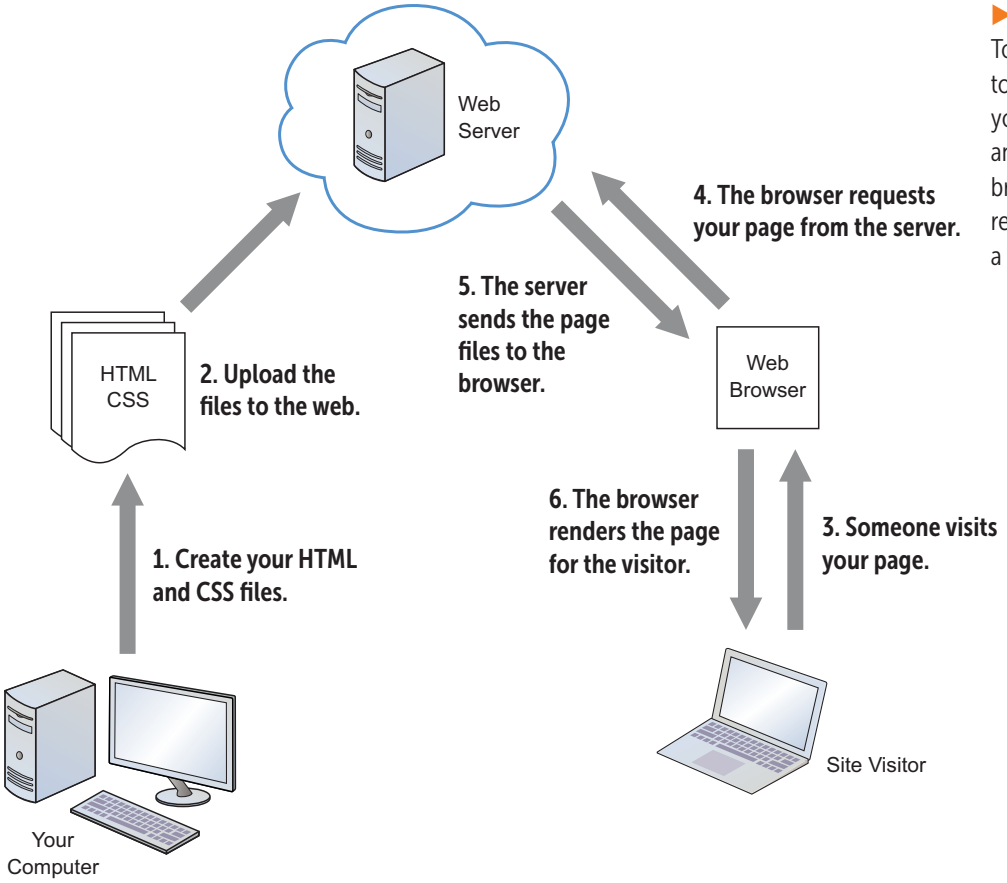
One of the most extraordinary facts about the web is that (with the exception of extra features such as images, videos, and sounds), its pages are composed of nothing but text. That's right—almost everything you see as you surf the web was created by stringing together the letters, numbers, and symbols that you can tap on your keyboard.

That idea is a mysterious one, to say the least, so I'll give you a quick look at how it works. Figure 1.4 shows the process.

The following steps explain the process in detail:

1. You use a text editor or similar software to create your HTML and CSS files.
2. You upload your HTML and CSS files to an online service called a *web hosting provider*, which runs a web server.

   When you sign up for an account, the hosting provider issues you a unique address, such as www.yourdomain.com. So if you upload a file named index.html, the address of that page is www.yourdomain.com/index.html.
3. A site visitor uses her web browser to type the address of your page.
4. The web browser uses that address to request your page from the web server.
5. After making sure that the address is correct, the web server sends the page to the user's web browser.
6. The web browser interprets the page's HTML tags and CSS properties through a process called *rendering*, and the rendered code appears on the user's device.

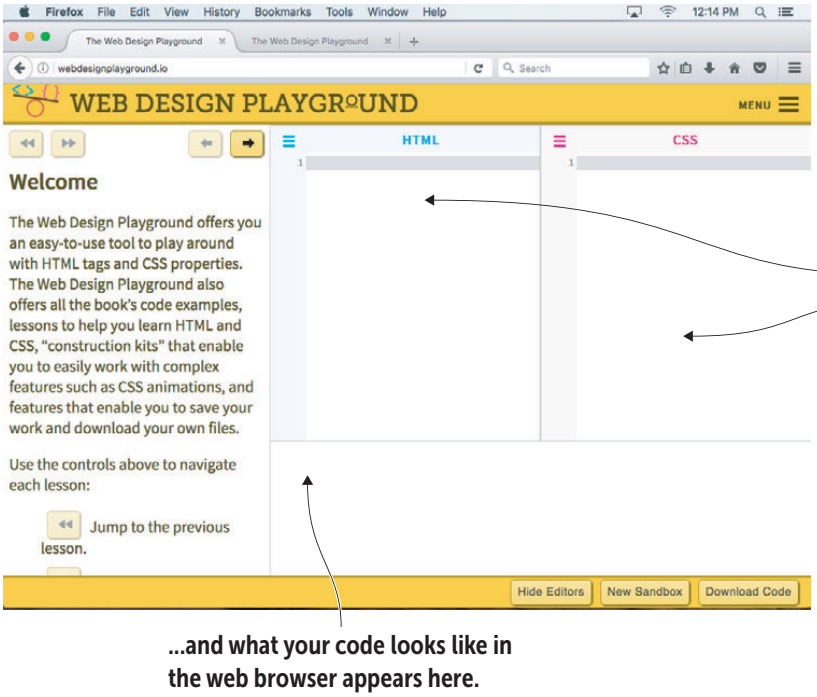


▶Figure 1.4
To go from HTML and CSS to a web page, you send your code to a web server, and visitors use their web browsers to retrieve and render your code into a page.

As you can see, the fact that the web is made of simple stuff doesn't mean that getting that stuff on the web is a simple matter. In fact, the procedure is a bit convoluted, especially when you're starting. That's why I devote appendix A to the process.

Introducing the Web Design Playground

Right now, though, you're probably itching to start playing around with HTML and CSS and seeing what these fascinating technologies can do. I don't blame you. One of this book's core ideas is that the best way to learn HTML and CSS is to have fun playing with your new knowledge, trying out different tags and properties, and experimenting with different values. To help you do all that with the least amount of fuss, I've built an online tool called the Web Design Playground, shown in Figure 1.5, which you can access at https://webdesignplayground.io/.



▶Figure 1.5
The Web Design Playground lets you play with HTML and CSS online.

You can use this site to try out HTML tags and CSS properties, load the book's example files, run through lessons that help you learn a topic, access various "construction kits" for experimenting with features, save your work, and even download the resulting file to your computer. The next few sections provide the details.

Playing with HTML and CSS

The main purpose of the Web Design Playground is to provide an easy-to-use tool for playing around with HTML tags and CSS properties. Here's how it works:

1. In the Web Design Playground, use the HTML Editor to type the HTML tags you want to try.

   If a tag requires one or more attributes, be sure to add them as well.
2. Use the CSS Editor to type the CSS property definitions you want to use.
3. Examine the Results box, which displays what your HTML and CSS will look like in a web browser.
4. Repeat steps 1–3 to fix any problems or perform further experiments.

Loading the Lesson Files

This book contains a ton of HTML and CSS code. As a general rule, you'll learn these subjects in a deeper way if you type the examples by hand

(which gives you what I call a "fingertip feel" for the code). I understand, however, that you're a busy person who may not have the time to type each example. To help you, the Web Design Playground includes a menu that links to every lesson from the book. When you select a lesson, you see an introduction followed by one or more examples and then by one or more activities that help you learn the lesson material. In each case, the code appears automatically, and you can play around with it as you see fit.

Here are the steps to follow to load a lesson:

1. In the Web Design Playground, click Menu at the right end of the toolbar. A menu of the site's links appears.

   The Book Lessons section contains an item for each chapter in the book.
2. Click the chapter that contains the lesson you're looking for.
3. In the submenu that appears, click the lesson you want to play with.

   The lesson introduction appears.
4. Click the Next Page button.

   The lesson example's HTML tags and text appear in the HTML Editor, and the example's CSS code appears in the CSS Editor.
5. Click Next Page to work through the activities for the lesson.
6. To jump to another lesson in the same chapter, click the drop-down menu above the Previous Page and Next Page buttons, and then click the lesson you want to see.

Preserving Your Work

You'll spend most of your time in the Web Design Playground performing experiments and trying out this book's exercises. Occasionally, however, you'll create some code that you want to save. The Web Design Playground gives you two ways to do that:

- *Copy some code.* To copy code for use elsewhere, use the HTML Editor or the CSS Editor to select the code you want to copy; click the editor's Menu icon; and then click Copy to Clipboard.
- *Download your work.* Click Menu, and below the Sandbox heading, click Download Code. This command saves the HTML and CSS and separate files, which are stored in a zip archive and downloaded to your web browser's default downloads folder.

Now that you know what you can do with HTML and CSS and how to use the Web Design Playground, you're ready to use the Playground to understand how to work with HTML tags and CSS properties.

Lesson 1.1: *Introducing HTML Tags*

Covers: HTML tags

PLAY

The addresses that appear here and elsewhere in this chapter refer to locations in the Web Design Playground, this book's companion online site. See "Introducing the Web Design Playground" earlier in this chapter.

Online: wdpg.io/1-1-0

HTML works its magic through short codes called *tags*. Each tag consists of three parts:

- An opening left angle bracket (<), also known as the *less-than sign*.
- The name of the element you want to use. Element names are short alphanumeric codes such as `p` for a paragraph, `em` for emphasis, and `h1` for a first-level heading.
- A closing right angle bracket (>), also known as the *greater-than sign*.

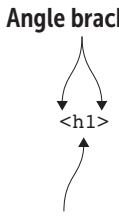


Figure 1.6
The structure of a typical HTML tag

In most cases, the tag tells the browser to start laying out the page according to the element you specified. If you add the `<em>` tag, for example, you're telling the browser to display the text that follows in italics. (em is short for *emphasis*.) You also have to tell the browser when you want it to stop displaying the text with that element, so you need to add a companion called the *closing tag*. (The original tag is the *opening tag*.) The closing tag is the same as the opening tag except that it requires a forward slash before the element name. A closing tag consists of the following four parts:

- An opening left angle bracket (<)
- A forward slash (/)
- The name of the element
- A closing right angle bracket (>)

MASTER

Throughout this book, I use the word element *to refer to a specific item of HTML, such as* `p` *or* `em`*, and the word* tag *to refer to the element and its surrounding angle brackets, such as* `<p>` *or* `<em>`*.*

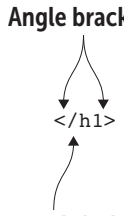


Figure 1.7
The structure of the closing tag for the h1 element

Figure 1.7 shows the closing tag for an `h1` element.

Together, the opening and closing tags create a kind of container to which you add some text (or even other elements); the browser displays the text according to the element that you specify in the tags. In Figure 1.1 earlier in this chapter, the text *How New Words Are Created* appears at the top of the file. To turn that text into the article's main heading as shown in Figure 1.2, I applied the `<h1>` tag, which displays the text as a first-level heading. The following example shows how I did it.

▶ Example ⇨ **Online:** wdpg.io/1-1-1

This example uses the `h1` *element to turn the text How New Words Are Created into a first-level heading.*

WEB PAGE

How New Words Are Created ← Text rendered as an h1 heading

HTML

```
<h1>How New Words Are Created</h1>
```

The opening tag — The affected text — The closing tag

By adding a few characters, you're telling the browser to do a whole bunch of things to the text:

- Display the text in its own paragraph.
- Add a bit of vertical space above and below the text.
- Format the text as bold.
- Format the text larger than the regular-page text to make clear that the text is a heading.

You learn more about headings in Chapter 2, but you can see that this deceptively simple code lets you do many things without much work. That's the magic of HTML.

Adding HTML Tag Attributes

Many HTML elements require no embellishment: You add the tag to the page, and the browser does the rest. A few tags, however, do require extra information before the web browser can process them correctly. You use the `<img>` tag, for example, to insert a picture into a web page, but you need to tell the web browser where your image is located. Similarly, to create a link, you use the `<a>` tag, but again, the web browser needs more info. In this case, it needs to know *what* you want to link to (such as the address of another website).

You supply these and similar extra bits of data to the browser by adding one or more attributes to the tag. An *attribute* is a name-value pair in which the name tells the browser the specific attribute and the value assigns it the particular setting you want to use.

MASTER

Although most HTML elements have both an opening and a closing tag, not all of them do. The element that you use to insert an image, for example (see Chapter 6), doesn't require a closing tag. These tags are known as self-closing tags.

PLAY

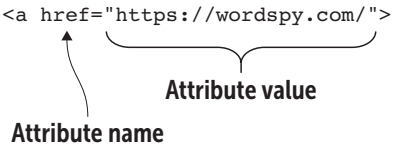
The text in Figure 1.1 has several single-word paragraphs that are intended to be headings. Line 7, for example, consists of the text Combining. *Given what you've learned about applying a first-level heading to the article title, apply a second-level heading to the* Combining *text.*

⇨ **Online:** wdpg.io/1-1-3

REMEMBER

Although technically, you're allowed to mix lowercase and uppercase letters in HTML element names and attribute names, I highly recommend using only lowercase letters. All-lowercase is the norm in web design because it's easier to type and read. You should also use lowercase for attribute values except when a specific value requires some uppercase, such as in a filename or an address.

When you're writing a link, for example, you specify the link address by adding the `href` attribute and setting its value to the address you want to use. Figure 1.8 shows an example.



▶ **Figure 1.8**
You can use attributes to specify extra data for some HTML elements, such as the link address for an `<a>` tag.

Here, the `href` (short for *hypertext reference*) attribute is assigned the value `https://wordspy.com/`, which is the address the user will be taken to if she clicks this link. Notice that the attribute value is surrounded by double quotation marks. These quotation marks are optional, but using them makes your code easier to read and maintain.

When combined with attributes, HTML can do some useful, powerful things. But HTML isn't the only web page tool you get to play with. In many ways, CSS is far more powerful and fun than HTML, and you begin learning how it works in the next section.

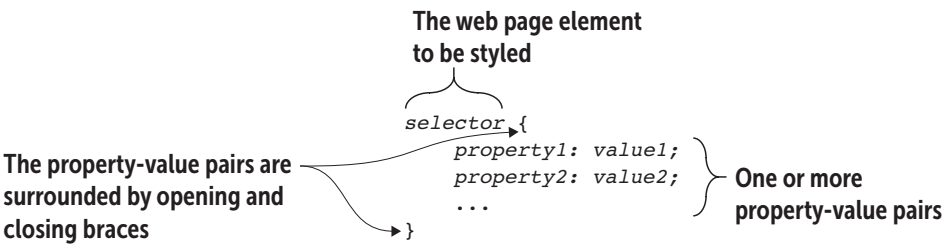
Lesson 1.2: *Introducing CSS Properties*

Covers: CSS properties

Online: wdpg.io.com/1-2-0/

CSS consists of a large collection of items called *properties* that control aspects of your page such as the text color, the font size, and the margins that surround an object. For each property you want to use, you assign a value, and that property-value pair (also known as a *declaration*) is the instruction that the browser carries out.

You have multiple ways to define a style, as you see in Chapter 7. For now, I'll go through the two most common methods. Figure 1.9 shows the general form of the first method.



▶ **Figure 1.9**
The syntax to use for defining CSS properties

From Figure 1.9, you see that defining a style consists of the following five parts:

- A reference to the web page element or elements to which you want the style applied. This reference is known as a *selector* because you use it to choose which page elements you want the browser to style.
- An opening left brace ({).
- The name of the property you want to apply. Property names are short alphabetic codes such as `color` for the text color, `font-size` for the text size, and `margin` for the margin size. The property name is always followed by a colon (`:`) and then a space for readability.
- The value you want to assign to the property, as well as the unit you want to use, if necessary. To specify a text size in pixels, for example, you add `px` to the value. The value is always followed by a semicolon (`;`).
- A closing right brace (}).

Taken together, these five parts comprise a style *rule*. The following example shows the style rule I used to tell the browser to set the font size for the main (`h1`) heading in Figure 1.2.

▶ *Example* ⇨ **Online:** wdpg.io/1-2-1

This example uses CSS to apply the `font-size` *property to the* `h1` *element.*

WEB PAGE

How New Words Are Created ← Text rendered with a 36-pixel type size

CSS

```
h1 {
    font-size: 36px;
}
```

The item you want to style and the opening brace

The closing brace

The style property and its value

HTML

```
<h1>How New Words Are Created</h1>
```

PLAY

How would you format a web page's second-level headings with a font size of 30 pixels? ⇨ **Online:** wdpg.io/1-2-2

The style begins by referencing the `h1` HTML element, which tells the browser to apply what follows to every `<h1>` tag in the current web page. After the opening brace ({), the next line specifies the property-value pair: `font-size: 36px;`. This line instructs the web browser to display every instance of `h1` text at a font size of 36 pixels. Finally, the closing brace (}) completes the style rule.

Here, you see one of the great advantages of using styles. If your page has a dozen `h1` headings, this rule applies to them all, which gives the page a consistent look. Even better, if you decided that a size of `48px` would look nicer for your headings, you'd have to change the value only once in the style rule, and that change would get reflected automatically in all your `h1` headings.

Note that you're not restricted to a single declaration in your style definitions. As you can see in the following example, you can add multiple declarations as needed.

▶ Example

⇨ **Online:** wdpg.io/1-2-3

This example uses specifies multiple properties in a single CSS definition.

WEB PAGE

CSS

```
h1 {
    font-size: 36px;
    text-align: center;
    font-variant: small-caps;
}
```

This property centers the heading.

This property displays the heading in small capital letters.

The closing brace

HTML

```
<h1>How New Words Are Created</h1>
```

Here, I've added the declarations `text-align: center;` to center the heading and `font-variant: small-caps;` to display the heading in small capital letters.

I mentioned earlier that you have another way to specify a style. You can insert the declaration directly into an HTML element by using the `style` attribute:

```
<element style="property1: value1; property2: value2; etc.">
```

Here's an example:

```
<h1 style="font-size: 36px; text-align: center">
```

When you use this method, your styles apply only to the HTML element in which they're declared. I talk more about this method in Chapter 7.

CSS is slightly more complicated than HTML, but with that complication comes immense power and expressiveness. As you see throughout the rest of this book, CSS is your royal road to creating fantastic, fun web pages.

When your HTML structure is festooned with CSS formatting, you can create beautiful web pages that are a pleasure to read and navigate.

MASTER

In this section's examples, I used four spaces to indent the declarations. This indentation isn't required, but it makes CSS much easier to read, so it's a good idea to get into the habit of indenting your properties.

Some Helpful Features of the Playground

Now that you know what HTML tags and CSS properties look like, you can return to the Web Design Playground and run through a few features that are designed to help you enter your tags and properties correctly:

- The HTML tags and CSS property names and values appear in colors that are different from the regular text. These colors help you differentiate between code and noncode.
- In the HTML box, when the text cursor is inside a tag, the editor automatically highlights both that tag and its companion tag. In Figure 1.10, you see that when I have the cursor in the opening `<p>` tag (which is the tag for creating a paragraph—see Chapter 2), the editor highlights that tag as well as its closing `</p>` tag. This highlighting gives you a visual indicator that you've closed your tags.

PLAY

How would you format a web page's second-level headings with a font size of 30 pixels and right alignment?

⇨ Online: wdpg.io/1-2-4

▶ **Figure 1.10**
The Web Design Playground's HTML editor highlights both the opening and closing tags when the cursor is inside one of them.

- The CSS editor has a similar feature: When the cursor is immediately to the left or right of a brace, the editor highlights the companion brace. This highlighting helps you make sure to enter both the opening and closing braces when you define a style.

- You can adjust the relative sizes of the editors by dragging the vertical border that separates the editors.
- The Web Design Playground can do a limited amount of error checking if you click an editor's Menu icon (pointed out in Figure 1.10) and then click Display Errors. If the editor detects something wrong, you see a red error indicator in the margin to the left of the line that has the problem. Hovering the mouse pointer over that icon displays the error message. If you forget the forward slash in a closing tag, for example, you see the error `Tag must be paired`, as shown in Figure 1.11.

▶ **Figure 1.11**
If the Web Design Playground detects a problem, an error icon appears in the margin to the left of the code, and hovering the mouse over the icon displays the error message.

Summary

- HTML defines the structure of your web page, whereas CSS defines the presentation.
- An *HTML tag* is a short code surrounded by angle brackets—such as `<h1>` or `<p>`—that applies an effect or inserts an object. Most tags also require a closing tag, such as `</h1>` or `</p>`.
- A *CSS property* is a name-value pair, and a CSS definition (or rule) is one or more properties surrounded by braces ({ and }) applied to a specified element (such as a tag name).
- To see this book's lessons and to play around with HTML and CSS code, use this book's companion website, the Web Design Playground: https://webdesignplayground.io/.