CHAPTER 3

Cascading Style Sheets (CSS)

CHAPTER OBJECTIVES

- Understand the philosophy of how HTML and CSS should fit together.
- Know when to use the different selectors—type selectors, the universal selector, class selectors, ID selectors.
- Know the syntax for the different selectors.
- Apply CSS rules to the `span` and `div` elements.
- Understand the different cascading levels for CSS rules.
- Implement external CSS files.
- Understand the syntax and meaning of the color properties and their values.
- Understand the syntax and meaning of the font properties and their values.
- Understand the syntax and meaning of the text properties and their values.
- Understand the syntax and meaning of the border properties and their values.
- Understand the syntax and meaning of the `padding` and `margin` properties and their values.

© StationaryTraveller/Getty Images

CHAPTER OUTLINE

3.1 Introduction

In the last chapter, we focused primarily on how to implement web page content. In this chapter, we focus on presentation of web page content. As you may recall, presentation refers to appearance and format. If you think appearance and format aren't all that important, think again. If your web page doesn't look good, people might go to it, but they'll leave quickly. An early exit might be OK if you're helping Grandma post her cat videos, but it's unacceptable for a business trying to generate revenue.

In this chapter, we start with an overview of Cascading Style Sheets (CSS) concepts and CSS basic syntax. We put those things into practice by applying CSS rules to various elements, including `span` and `div` elements. We show you how to position those rules (1) at the top of the web page's main file or (2) in an external file. In the second half of the chapter, we describe *CSS properties*. Properties are the hooks used to specify the appearance of the elements within a web page. Specifically, we introduce CSS properties for color, font, and line height. Also, we introduce CSS properties for borders, padding, and margins.

3.2 CSS Overview

The W3C's philosophy in terms of how HTML and CSS should fit together is (1) use HTML elements to specify a web page's content, and (2) use CSS to specify a web page's appearance. There are lots and lots of CSS properties that enable you to determine a web page's appearance. In this chapter, we'll cover quite a few of those properties, but not even close to all of them. When

implementing a web page, if you need a particular format for an element and you can't find an appropriate CSS property in this book, don't give up right away. Search the Web for additional CSS properties to see if you can find one that suits your needs.

As you'll see shortly, and as you may recall from Figure 1.8's Kansas City Weather web page in Chapter 1, CSS code is normally separated from web page content code. Specifically, web page content code goes in the `body` container, whereas CSS code goes either at the top of the web page in the `head` container or in an external file. Why is that separation strategy a good thing? Because if you want to change the appearance of something, it's easy to find the CSS code—at the top of the web page or in an external file.

The current version of CSS is CSS3, and all major browsers support it. In 2009, the W3C started work on CSS4. There is no single, unified CSS4 specification. Instead, it's maintained as separate modules. Unfortunately, CSS4 is not fully supported by the major browsers yet. Thus, in this book, we stick with CSS3.

3.3 CSS Rules

The way CSS works is that CSS rules are applied to elements within a web page. Browsers determine which elements to apply the CSS rules to with the help of selectors. There are quite a few different types of selectors. For now, we'll introduce type selectors and the universal selector. Type selectors are very popular. The universal selector is not as popular, but it's important to understand it because you'll see it referred to on various websites, including the W3C's CSS website at https://www.w3.org/Style/CSS.

With a *type selector*, you use an element type (e.g., `hr`) to match all instances of that element type and then apply specified formatting features to those instances. For example, the following CSS rule uses a type selector with the `hr` element type and applies a width of 50% to all the `hr` elements in the current web page:

```
hr {width: 50%;}
```

A "width of 50%" means that for each `hr` element, its horizontal line will span 50% of the width of its enclosing container. Usually, but not always, the enclosing container will be the web page's `body` container.

Now for another type of selector—the universal selector. The *universal selector* uses the same syntax as the type selector, except that instead of specifying an element type, you specify `*`. The asterisk is a wildcard. In general, a *wildcard* is something that matches every item in a collection of things. For CSS selector rules, the `*` matches every element in a web page's collection of elements. Here's an example universal selector CSS rule that centers the text for every text-oriented element in the web page:

```
* {text-align: center;}
```

Even though the rule matches every element, because the property (`text-align`) deals with text, the rule affects only the elements that contain text.

3.4 Example with Type Selectors and the Universal Selector

Now let's look at a complete web page where we put into practice what's been covered so far in regard to CSS rules. Study the source code in FIGURE 3.1's Tree Poem web page. Notice the three CSS rules inside the `style` container. The first two rules should look familiar because they were presented in the previous section. The third rule uses a type selector with a slightly different syntax than before—there's a comma between two element types, `h2` and `p`. If you want to apply the same formatting feature(s) to more than one type of element, you can implement that with one rule, where the element types appear at the left, as part of a comma-separated list.

In Figure 3.1's three CSS rules, notice the four property-value pairs inside the { }'s, and copied here for your convenience:

- `text-align: center`
- `width: 50%`
- `font-style: italic`
- `color: blue`

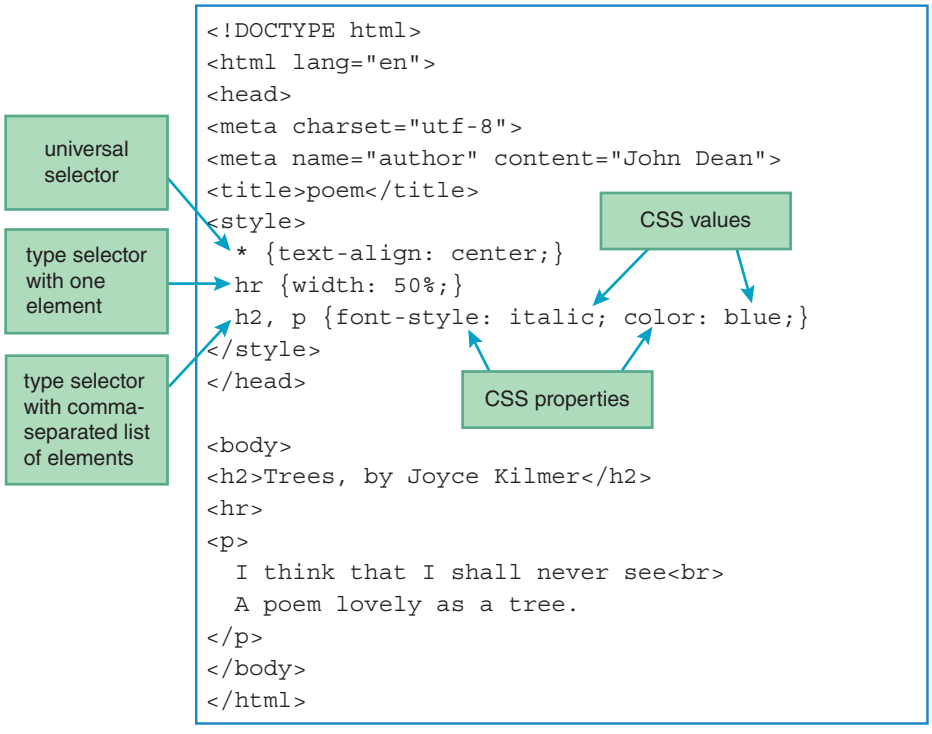


FIGURE 3.1 Source code for Tree Poem web page

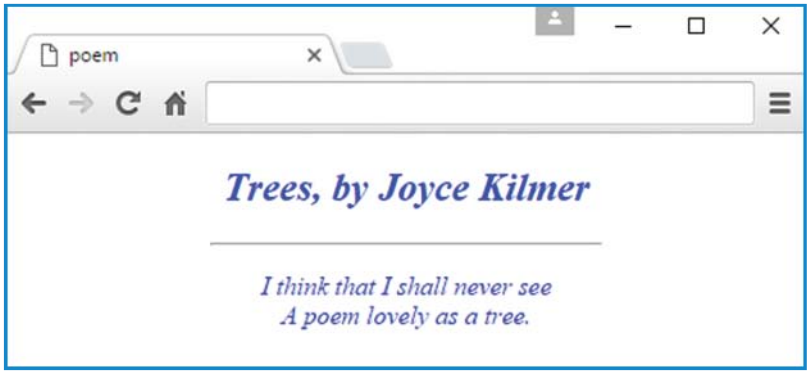


FIGURE 3.2 Tree Poem web page

We'll cover those properties in detail later on, but for now, go ahead and guess what they are for and how they affect the appearance of the Tree Poem web page. After you've made your guess, take a look at the resulting web page in FIGURE 3.2.

In the Tree Poem web page, the `* {text-align: center;}` rule causes the elements that contain text to be centered. The `hr` element does not contain text, so it's not affected by the `textalign` property. Nonetheless, as you can see, it's also centered. That's because `hr` elements are centered by default.

The `hr {width: 50%;}` rule causes the horizontal line to render with a width that's 50% of the web page `body`'s width.

Finally, the `h2, p {font-style: italic; color: blue;}` rule causes the heading and paragraph elements to be italicized and blue.

3.5 CSS Syntax and Style

CSS Syntax

In this section, we address CSS syntax details. First—the syntax for the `style` container. Refer back to Figure 3.1 and note how the three CSS rules are enclosed in a `style` container. Here's the relevant code:

```
<style>
  * {text-align: center;}
  hr {width: 50%;}
  h2, p {font-style: italic; color: blue;}
</style>
```

If you go back to the figure, you can see the `style` container positioned at the bottom of the web page's `head` container. It's legal to position it in the `body` container, but don't do it. Coding conventions suggest positioning it at the bottom of the web page's `head` container. By following that convention, other web developers will be able to find your CSS rules quickly.

In the `style` start tag, it's legal to include a `type` attribute with a value of `"text/css"`, like this:

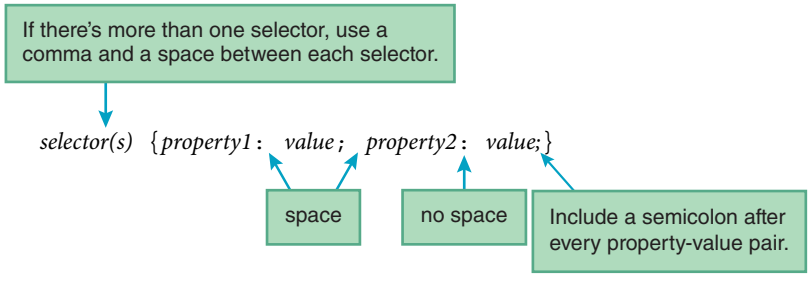
```
<style type="text/css">
```

In the Tree Poem web page, you can see that the `type` attribute is omitted. Currently, `text/css` is the only legal value for the `type` attribute, and it's the default value for the `type` attribute. So why did the HTML designers include a `type` attribute at all if there's only one type? They wanted to leave open the possibility of having different `style` types in the future. Google's Style Guide, which covers both HTML and CSS, recommends that you reduce the size of your web page file by omitting the `type` attribute, and we follow that convention in this book.

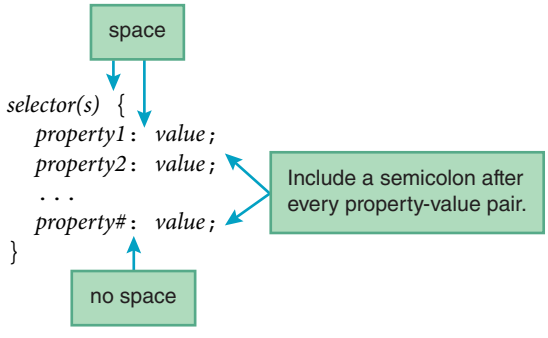
CSS Style

Now we'll look at some CSS guidelines that are not enforced by browsers or the HTML5 standard. They are style guidelines, and you should follow them so your code is easy to understand and maintain.

For short CSS rules, use this format:



Remember in Chapter 2 when we introduced block formatting for multi-line container elements? That's where the start tag and end tag are aligned at the left, and interior lines are indented. Block formatting for CSS rules is similar in that the first and last lines are aligned at the left, and interior lines are indented. If you have a CSS rule that's kind of long (at least two or three property-value pairs), you should use block formatting like this:



With both short and long CSS rules, the W3C CSS standard allows you to omit the semicolon after the last property-value pair. However, coding conventions suggest that you should not omit the last semicolon—you should include it. That way, if another property-value pair is added later on, there will be less likelihood of accidentally forgetting to add a semicolon in front of the new property-value pair.