Learning More About Styles

Style is a way to say who you are without having to speak. —Rachel Zoe

This chapter covers

- Learning the three methods for adding styles to a web page
- Adding power and flexibility with classes
- Understanding the units of measurement you can use in your CSS

How do you craft pages that rise above the humdrum? How do you design pages that go beyond the same old, same old? One word: ***styles***. If you've seen a web page that you think is well designed, know that the page uses styles to achieve that look. If there's a web designer whose work you admire, know that the designer mastered styles that make her work stand out. You saw several useful styles in Part 1 of the book, but those styles are only a taste of what's out there. To help you get started down the road to becoming truly style-savvy, this chapter takes your style knowledge to the next level.

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>

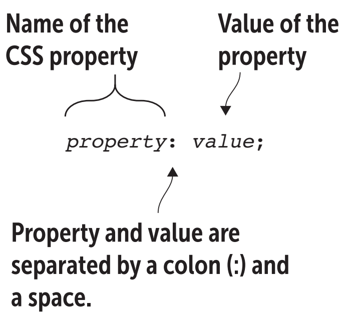
Learning More About Styles

Adding Styles to a Page

I mentioned in Chapter 1 that a web page is a text file filled with words, numbers, and a few strategically placed HTML tags that provide structure for the text. You'll be happy to hear that CSS is also a text-based business, so you don't need anything grander than a simple text editor (or this book's handy Web Design Playground) to get started with styles.

That said, although *what* styles consist of is simple enough, *how* you add styles to a web page is a bit more complex. First, recall from Chapter 1 that a single style declaration consists of a property-value pair that uses the syntax shown in Figure 7.1.

REMEMBER
A style is an instruction to the browser to modify how it displays something on the page, and a style sheet (the SS part of CSS) is a collection of styles. So throughout this book, I use the terms CSS and styles interchangeably.



▶ Figure 7.1
The syntax of a property-value pair

The `property` name is almost always written in lowercase letters (although it doesn't have to be). If the `value` includes one or more spaces, numbers, or punctuation characters other than a hyphen (-), surround the value with quotation marks.

The added complexity of CSS comes from the fact that you have not one, not two, but *three* ways to tell the web browser what style declarations you want to use:

- Inline styles
- Internal styles
- External styles

The next three lessons introduce you to these methods.

FAQ
Do I have to add a space after the colon? Strictly speaking, no, the space isn't required. I do recommend adding the space, however, because it makes your styles more readable.

Lesson 7.1: *Inserting Inline Styles*

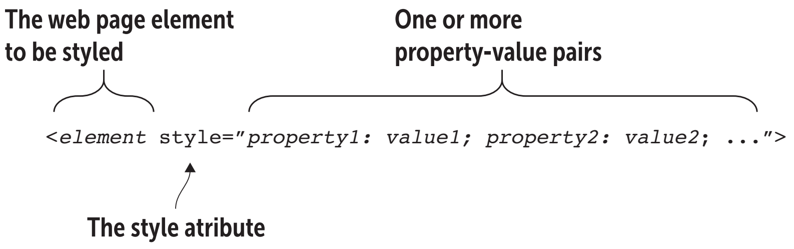
Covers: The `<style>` attribute

Online: wdpg.io/7-1-0

Probably the most straightforward way to add styles to your web page is to insert them directly into the element you want to modify. This technique is called an *inline style*, and you insert a style by including the `style` attribute within the HTML element you want to change. Figure 7.2 shows the general syntax to use.

USE IT
A good use for inline styles is to perform a quick test of a CSS property or two. If you're getting started with a page and haven't yet set up an external style sheet (see Lesson 7.3), inline styles are a quick way to test-drive some CSS.

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>


The web page element to be styled
One or more property-value pairs
<element style="property1: value1; property2: value2; ...">
The style atribute

▶ Figure 7.2 The syntax to use for inline styles

Here are a few points to keep in mind when you use inline styles:

- If you want to include two or more property-value pairs in a single inline style, be sure to separate each pair with a semicolon (;).
- If a value needs to be quoted, use single quotation marks (').
- An inline style affects only the element in which you place the `style` attribute.

Following are a couple of examples of inline styles.

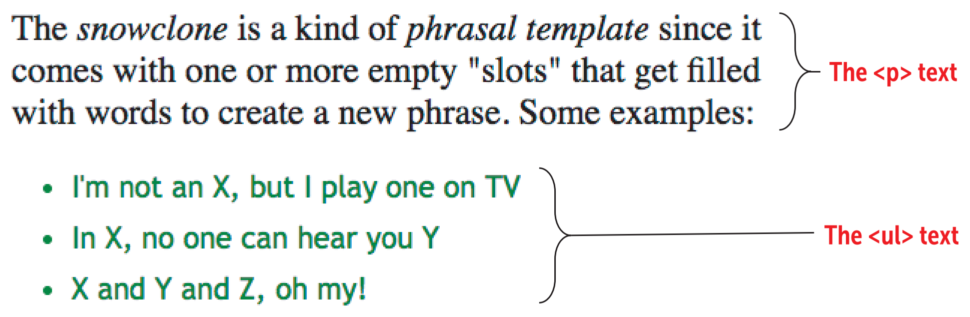
BEWARE

Because the `style` attribute's value is itself surrounded by double quotation marks, be careful if one of your property-value pairs requires quotation marks of its own. In that case, surround the value with single quotation marks (for example, `style="font-family: 'PT Sans';"`).

▶ ***Example*** ⇨ **Online:** wdpg.io/7-1-1

This example shows an inline style applied to a `<p>` tag, as well as an inline style with multiple property-value pairs applied to a `<ul>` tag.

WEB PAGE


The *snowclone* is a kind of *phrasal template* since it comes with one or more empty "slots" that get filled with words to create a new phrase. Some examples:
• I'm not an X, but I play one on TV
• In X, no one can hear you Y
• X and Y and Z, oh my!
The <p> text
The <ul> text

continued

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>

HTML

The p element's inline style sets the font size.

```
<p style="font-size: 1.5em"> The <i>snowclone</i> is a kind of
<i>phrasal template</i> since it comes with one or more empty
"slots" that get filled with words to create a new phrase. Some
examples:</p>
<ul style="color: darkgreen; font-family: 'Trebuchet MS',
sans-serif; font-size: 1.25em;">
    <li>I'm not an X, but I play one on TV</li>
    <li>In X, no one can hear you Y</li>
    <li>X and Y and Z, oh my!</li>
</ul>
```

The ul element's inline styles set the text color, typeface, and size.

PLAY

Can you spot the `style` *attribute error in the following* `<a>` *tag?* `<a href="https://www.w3.org/TR/css-style-attr/" style="color: indianred; font-weight: bold, text-decoration: none;">` ⇨ **Online:** wdpg.io/7-1-3

Although inline styles are the easiest way to add CSS code to your page, they're not the most convenient method for anything other than the simplest of pages because they require you to add the `style` attribute directly to every element you want styled. If your page consists of, say, a dozen `h2` elements, and you want to apply the same style to them all, you must add a dozen `style` attributes. Even worse, if you later decide to change how your `h2` elements appear, you have to change every instance of the `style` value. That's a lot of work, so most web designers eschew inline styles or use them only for specific instances.

What do these designers do instead? Ah, that's where internal styles come in.

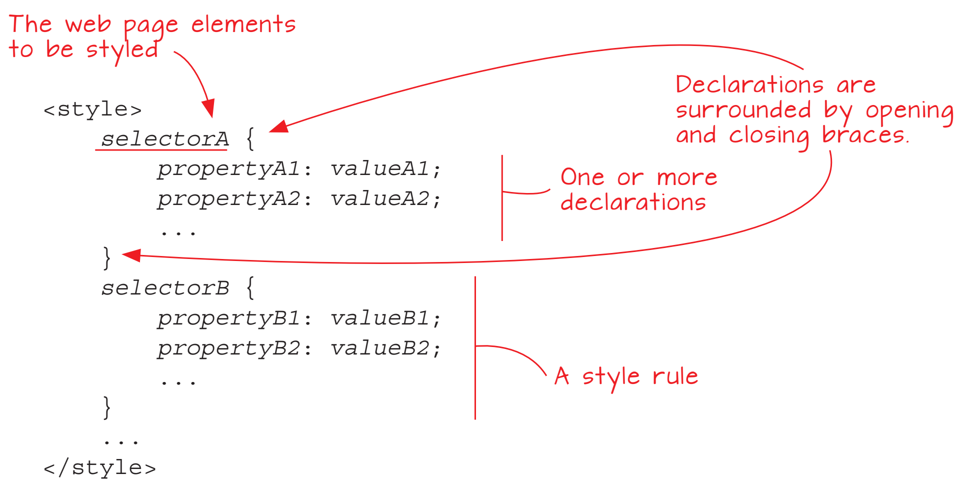
Lesson 7.2: *Adding an Internal Style Sheet*

Covers: The `style` element

⇨ **Online:** wdpg.io/7-2-0

The second method for getting styles into a web page involves adding a `<style></style>` tag pair in the page's head section (that is, between the page's `<head>` and `</head>` tags) and then defining the styles within those tags. This method is called an *internal style sheet* (or sometimes an *embedded style sheet*), and it uses the following general syntax:

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>


The web page elements to be styled
Declarations are surrounded by opening and closing braces.
One or more declarations
A style rule

```
<style>
    selectorA {
        propertyA1: valueA1;
        propertyA2: valueA2;
        ...
    }
    selectorB {
        propertyB1: valueB1;
        propertyB2: valueB2;
        ...
    }
    ...
</style>
```

From this syntax, you can see that an internal style sheet consists of one or more *style rules*, each of which defines one or more property-value pairs to be applied to the specified web page elements. Each rule has the following characteristics:

- A *selector* that specifies the web page elements to which you want the style applied. This selector is often a tag name, but it can also specify any other type of CSS selector (such as the class selector, described in Lesson 7.4).
- An opening left brace: {.
- One or more property-value pairs, separated by semicolons.
- A closing right brace: }.

In CSS lingo, a property-value pair is called a *declaration*, and the collection of declarations applied to a selector—that is, the braces and the property-value pairs between them—is called a *declaration block*. The combination of a selector and its declaration block is called a *style rule*.

The following example uses an internal style sheet to format the `dt` element.

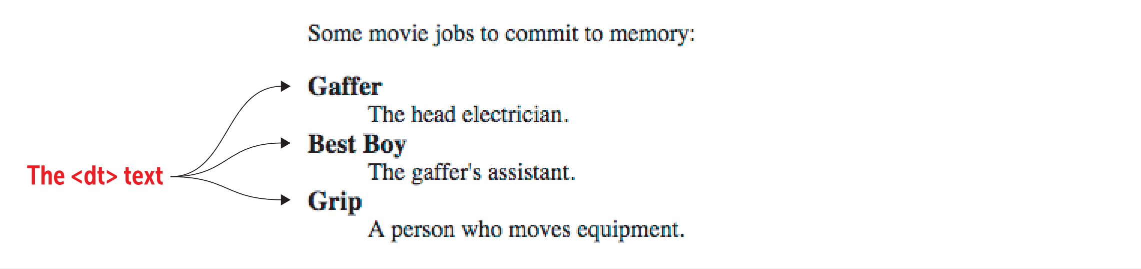
Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>

Example

Online: wdpg.io/7-2-1

This example uses an internal style sheet to apply a font size and bolding to each of the `<dt>` *tags.*

WEB PAGE


Some movie jobs to commit to memory:

Gaffer
The head electrician.
Best Boy
The gaffer's assistant.
Grip
A person who moves equipment.

CSS

```
<style>
    dt {
        font-size: 18px;
        font-weight: bold;
    }
</style>
```

The CSS rule for the dt element

HTML

```
<p>Some CSS jargon to commit to memory:</p>
<dl>
    <dt>Gaffer</dt>
    <dd>The head electrician.</dd>
    <dt>Best Boy</dt>
    <dd>The gaffer's assistant.</dd>
    <dt>Grip</dt>
    <dd>A person who moves equipment.</dd>
</dl>
```

The `<dt>` tags and text

MASTER

Declaration blocks can get quite long, with some containing a dozen or more property-value pairs. One way to make reading and working with these big blocks easier is to add the declarations in alphabetical order by property name.

Here, you see one of the great advantages of using internal styles. If your page has a dozen `dt` elements, this one style applies to them all, which gives the page a consistent look. Even better, if you decided that a size of `20px` would look better for your `dt` text, you'd have to change the value only once in the style declaration; that change would get reflected automatically in all your `dt` elements.

Internal styles work beautifully if your site consists of a single web page. Such sites aren't rare, but it's far more likely that your or your client's site will consist of several pages, perhaps even several dozen. If you want your pages to have a consistent look—and you should, because consistency

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>

across pages is one of the hallmarks of good web design—using internal style sheets means copying the same `<style>` tag to each and every page. Also, if you change even one aspect of any style rule, you must make the same change to the same rule in every page.

The bigger your site is, the less appealing all that maintenance sounds and the more likely you'll be to switch to external style sheets.

Lesson 7.3: Referencing an External Style Sheet

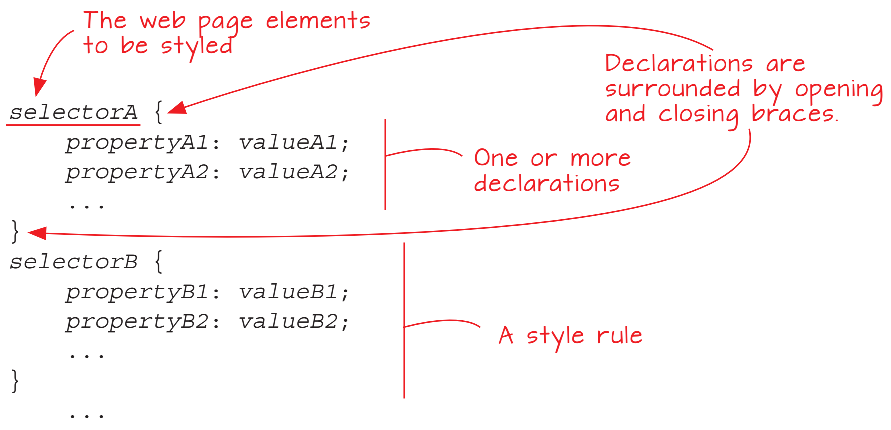
Covers: The `link` element

Online: wdpg.io/7-3-0

The third and final method for adding styles to a page involves creating a second text file that you use to define your style rules. This method is called an *external style sheet*, and by tradition, its filename uses the `.css` extension (as in `styles.css`). Within that file, you use the same syntax that you saw earlier for an internal style sheet, but you do without the `style` element:

REMEMBER

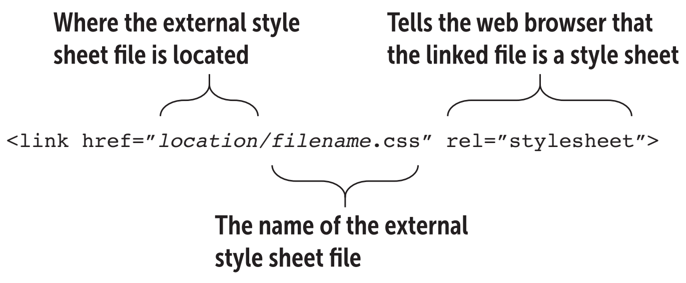
Traditionally, you save an external style sheet text file with the `.css` *extension* (`styles.css`)*.*



The web page elements to be styled
Declarations are surrounded by opening and closing braces.
One or more declarations
A style rule

```
selectorA {
    propertyA1: valueA1;
    propertyA2: valueA2;
    ...
}
selectorB {
    propertyB1: valueB1;
    propertyB2: valueB2;
    ...
}
    ...
```

To let the web browser know that you have an external style sheet, you add a `<link>` tag to your web page's head section. Figure 7.3 shows the syntax.



Where the external style sheet file is located
Tells the web browser that the linked file is a style sheet
The name of the external style sheet file

```
<link href="location/filename.css" rel="stylesheet">
```

Figure 7.3 The `<link>` tag syntax for attaching an external style sheet to a web page

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>

In this syntax, the *`location`* value is perhaps the trickiest. There are four possibilities:

- *Referencing a CSS file in the same directory.* Leave out the location and reference only the filename, like so:

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

- *Referencing a CSS file in a subdirectory of the web page directory.* The location is the name of the subdirectory. If the subdirectory is named `css`, for example, you'd use the following:

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

- *Referencing a CSS file in a subdirectory of the website's main subdirectory.* The location is the root directory (/) followed by the name of the subdirectory. If the subdirectory is named `css`, for example, you'd use the following:

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

- *Referencing a CSS file on a remote server.* The location is the full URL of the CSS file. Here's an example:

```
  <link href="https://fonts.googleapis.com/css?family=Lato"
rel="stylesheet">
```

REMEMBER

As with the `<style>` tag, you may see some CSS external file `<link>` tags that include the `type="text/css"` attribute. That attribute was required with HTML 4.01, but you don't need it with HTML5.

Using an external style sheet brings three major advantages to your web pages:

- *It makes applying a consistent look across multiple pages much easier.* If you attach the same external style sheet to several pages, and that CSS styles, say, your `h1` elements, those tags will look exactly the same on all the pages.
- *It makes updating and maintaining your pages much easier.* If you make a change to the CSS in an external style sheet, that change is automatically propagated to every web page that links to the CSS file.
- *It enhances the separation between structure and presentation.* By using an external style sheet, you separate your project into two distinct layers: a *structural layer* of files that contain only HTML tags and a *presentation layer* of files that contain only CSS rules. Nice.

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>

This isn't to say that you should use only external style sheets rather than inline styles or internal style sheets. You have plenty of good reasons to use the `style` element, and you'll find that some web-page design problems are most easily solved by using a `style` attribute in an HTML tag. There's no need for taking a dogmatic approach to CSS; do what works.

Lesson 7.4: ***Using Class Selectors***

Covers: The `.class` selector

Online: wdpg.io/7-4-0

Earlier, you learned that when you're defining a style rule, the first thing you specify is the web page object you want styled, followed by the declaration block:

```
selector {
    property1: value1;
    property2: value2;
    ...
}
```

The specified object is called a *selector*, and so far in this book, you've seen it used only with tag names, such as `h1` and `div`. This selector is known as the *type selector* because it targets a specific type of HTML element.

Type selectors are handy, and you'll use them frequently in your web-design career, but it doesn't take long before you come across a conundrum: What are you supposed to do when you have multiple instances of the same element that need different styling? A web page can easily have a few dozen `<div>` tags, so what's a coder to do if some of those `div`s require, say, right-aligned, italic, light gray text set at 20px and others require centered, bold, dark gray text set at 24px? You could insert all these styles as inline styles, sure, but that task quickly gets unwieldy when you're working with more than a half dozen elements.

You work around this and similar problems by taking advantage of the many other types of CSS selectors available. CSS derives most of its tremendous flexibility and power through these selectors. I don't think I'm exaggerating in the least when I say that if you want to become a CSS wizard—or (which is sort of the same thing) if you want to make yourself irresistibly hirable as a web designer—mastering selectors is the royal road to that goal. To get started down that road, check out perhaps the most powerful CSS selector: the class selector.

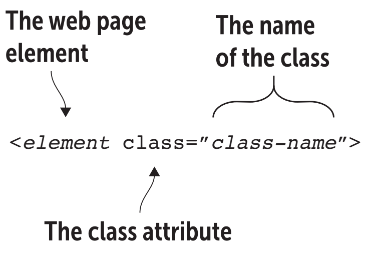
One of the most common web design scenarios is having multiple page objects that require the same styling. Whenever you have a set of elements that require the same styling, you can group those elements under a single HTML-and-CSS umbrella. In HTML, that umbrella takes the form of the `class` attribute, and the syntax appears in Figure 7.3.

REMEMBER

Although exceptions occur, for purposes of this book, your class names must begin with a letter; the rest of the name can include any combination of letters, numbers, hyphens (-), and underscores (_). See wdpg.io/7-4-3/.

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>

▶ Figure 7.4
Use the `class` attribute to assign a class name to an HTML element.



BEWARE
Class names are case-sensitive, meaning that the browser treats, say, `myClassName` *and* `myclassname` *as two separate classes.*

The following code assigns the class name `custom-bullet-text` to a `<span>` tag:

```
<span class="custom-bullet-text">
```

The key point here—and the source of the power inherent in using classes—is that you can assign the same class to multiple elements. When that's done, you can use an internal or external style sheet to define the styles for that class by using the class name, preceded by a dot (`.`) as the selector in your CSS:

```
.class-name {
    property1: value1;
    property2: value2;
    ...
}
```

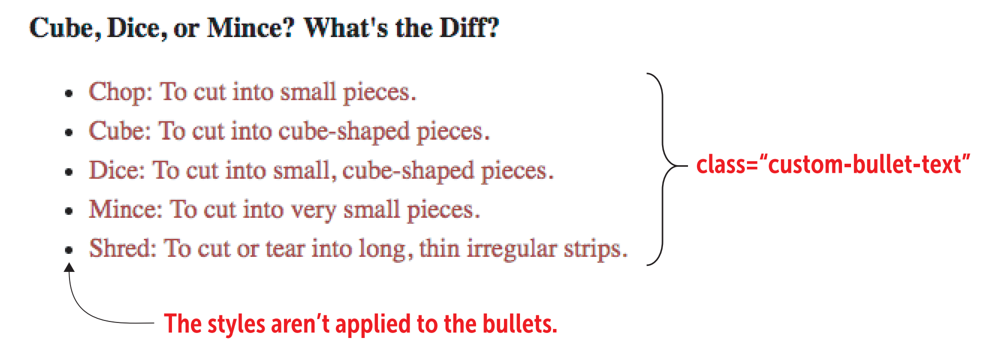
The following example shows you how to use a class selector.

▶ *Example*

⇨ **Online:** wdpg.io/7-4-1

This example assigns a class name to each `<span>` *tag and then uses a CSS class selector to apply a rule to those* `span` *elements.*

WEB PAGE



Cube, Dice, or Mince? What's the Diff?

- Chop: To cut into small pieces.
- Cube: To cut into cube-shaped pieces.
- Dice: To cut into small, cube-shaped pieces.
- Mince: To cut into very small pieces.
- Shred: To cut or tear into long, thin irregular strips.

class="custom-bullet-text"
The styles aren't applied to the bullets.

CSS

```
.custom-bullet-text {
    color: brown;
    font-size: 18px;
    line-height: 1.5;
}
```

Rule for the custom-bullet-text class

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>

HTML

```
<h3>Cube, Dice, or Mince? What's the Diff?</h3>
<ul>
   <li><span class="custom-bullet-text">Chop: To cut into small
pieces.</span></li>
   <li><span class="custom-bullet-text">Cube: To cut into cube-
shaped pieces.</span></li>
   <li><span class="custom-bullet-text">Dice: To cut into small,
cube-shaped pieces.</span></li>
   <li><span class="custom-bullet-text">Mince: To cut into very
small pieces.</span></li>
   <li><span class="custom-bullet-text">Shred: To cut or tear
into long, thin irregular strips.</span></li>
</ul>
```

The `custom-bullet-text` class is assigned to each `span` element.

Units of Measurement in CSS

Many web page styles require measurement values, including font sizes, border widths, and margin sizes. So far in this book, I've used pixels (`px`) to specify measurements, but you can use several other units, which I've laid out in Table 7.1.

MASTER

Why not apply the CSS to the `li` element in this example? Such a rule would also style the bullet. By wrapping each list item in a `<span>`, you can style only the text.

▶ **Table 7.1** Units of Measurement for CSS Properties

Unit	Name	Description
`px`	pixel	An absolute measurement equal to 1/96 of an inch
`pt`	point	An absolute measurement equal to 1/72 of an inch
`em`	em	A relative measurement equal to the element's default, inherited, or defined font size
`rem`	root em	A relative measurement equal to the font size of the root element of the web page
`vw`	viewport width	A relative measurement equal to 1/100 of the current width of the browser window
`vh`	viewport height	A relative measurement equal to 1/100 of the current height of the browser window

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>

BEWARE

Don't confuse the `em` *unit of measurement with the* `em` *element used to emphasize text in HTML.*

REMEMBER

The root element of a web page is the `html` *element. This element is automatically assigned either the browser's default type size (usually 16px) or the type size set by the user in the browser's preferences.*

PLAY

Classes are even more powerful than I've shown here, because you can apply multiple classes to a single element by separating class names with a space in the `class` *attribute value. The code* `<span class="red-text big-text">`*, for example, applies both the* `red-text` *class and the* `big-text` *class to a* `span` *element.*
⇨ **Online:** wdpg.io/7-4-2

Table 7.1 lists two types of units: absolute and relative. *Absolute* measures have a fixed size—a pixel is a pixel, for example—so you can be sure that an element sized with an absolute measure always appears consistently. As a designer, you may think this fact is a good thing, but it isn't always—especially on the web, where users sometimes change the default size of text in their browser settings. As a designer, your job should be to honor that change, not override it. Absolute values are frowned upon because they overrule type size changes set by the user, which is a design no-no. Also, as you'll see in Chapter 14, absolute values make your page design too rigid, so it doesn't show up well on both large and small screens.

Therefore, modern web-design best practices eschew absolute units in favor of relative units, usually rems or percentages. Relative measures don't have a fixed size. Instead, they're based on whatever size is supplied to the element. This size could be inherited from the parent element, or it could be the default specified by the user. If the browser's default type size is 16px, and you set your `<p>` type to `1.5rem`, your paragraph text will be rendered at 24px. If the user bumped up the default text size to `20px`, your paragraphs will render at 30px, thus preserving the relative size of the text. Also, relative measures scale well on devices of different sizes, so a design that looks good on a desktop screen can be made to look as good on a smartphone screen. (Again, Chapter 14 is the place to get the details.)

Summary

- Inline styles are added directly to a tag using the `style` attribute.
- You create an internal style sheet by adding your definitions to the `<style>` tag.
- An external style sheet exists as a separate `.css` file and is referenced through a `<link>` tag.
- A *class selector* applies CSS rules to any element that uses the specified class name.
- For CSS properties that require measurement values, use one of the following units: `px`, `pt`, `em`, `rem`, `vw`, or `vh`.

Licensed to Alessandro Provetti <alessandro.provetti@unimi.it>