mysite.com. To select links that point outside your site, you'll want to select all absolute links that *don't* point to the domain *mysite.com*. Here's how you would do that:

```
a[href^="http://"]:not([href^="http://mysite.com"])
```

Translated into English, this selector says "Select all links whose href attribute begins with *http://*, but *not* ones that begin with *http://mysite.com*." As you'll recall from page 60, in an attribute selector, ^= means "begins with." A shorter way to write the same thing would be this:

```
a[href^="http://"]:not([href*="mysite.com"])
```

In an attribute selector, *= means "contains," so this line would exclude any absolute URL that contains mysite.com. This would include *http://www.mysite.com* as well as *http://mysite.com*.

There are some limitations to the `:not()` selector:

- You can only use *simple selectors* with the `:not()` selector. In other words you can use element selectors (like `html` or `p`), the universal selector (* [see page 49]), classes (`.footer`, for example), IDs (`#banner`, for example), or pseudo-classes (`:hover`, `:checked`, `:first-child`, and so on). So the following are all valid:

```
.footnote:not(div)
img:not(.portrait)
div:not(#banner)
li:not(:first-child)
```

- You can't use descendant selectors (like `div p a`, pseudo-elements (like `::first-line`), group selectors, or combinators (like the adjacent sibling selector `h2 + p`).
- You can't string multiple `:not()` selectors together. For example, the following is invalid:

```
a[href^="http://"]:not([href*="google.com"]):not([href="yahoo.com])
```

In other words, you can only use `:not()` once with a selector.

Tutorial: Selector Sampler

In the rest of this chapter, you'll create a variety of selector types and see how each affects a web page. This tutorial starts with the basic selector types and then moves on to more advanced styles.

To get started, you need to download the tutorial files located on this book's companion website at *https://github.com/sawmac/css_mm_4e*. Click the tutorial link and download the files. All of the files are enclosed in a zip archive, so you'll need to unzip them first. The files for this tutorial are contained inside the folder named *03*.

INDIGNANTLY ASKED QUESTION

Keeping It Internal

Hey, what's up with the internal style sheet in this tutorial? Chapter 2 recommends using external style sheets for a bunch of reasons.

Think you're pretty smart, eh? Yes, external style sheets usually make for faster, more efficient websites, for all the reasons mentioned in Chapter 2. However, internal style sheets make your life easier when you're designing a single page at a time, as in this tutorial. You get to work in just one web page file instead of flipping back and forth between the external style sheet file and the web page.

Furthermore, you can preview your results without constantly refreshing your browser's cache; flip back to the box on page 25 for more on that quirkiness.

So, yes, you should use external style sheets for your sites. And if you were going to use the styles you created in this tutorial on more than just the single tutorial HTML file, you would. But just to keep things fast and simple as you learn CSS, you'll use a single HTML file and an internal style sheet for this exercise.

But, that was an excellent question! Keep up the good work.

1. **In your favorite text editor, open *03→selector_basics.html*.**

   This page is made of very basic HTML tags (see Figure 3-10). But you'll liven things up in this tutorial. First, you'll link to a Google font—the same one you used back on page 36.

2. **In the empty line below the closing `</title>` tag, type:**

```
<link href='http://fonts.googleapis.com/css?family=Varela+Round'
rel='stylesheet'>
```

   As described on page 36, this tag links to an external style sheet that Google hosts on its web servers. It downloads the Varela Round font and so you can use it on the page. (You'll learn more about using web fonts like this one from Google on page 140.) Next, you'll add the internal style sheet.

3. **After the `<link>` tag you added in the last step, hit Return and type `<style>`. Press Enter twice and then type `</style>`.**

   These are the opening and closing style tags—it's a good idea to type both tags at the same time, so you don't accidentally forget to add the closing `</style>` tag. Together, these two tags tell a web browser that the information between them is Cascading Style Sheet instructions. The HTML should now look like this (the stuff you added is in bold):

```
<title>Selector Basics</title>
<link href='http://fonts.googleapis.com/css?family=Varela+Round'
rel='stylesheet'>
<style>

</style>
```

TORIAL:
LECTOR
MPLER

Type selectors—like the one you're about to create—are the most basic kind of selector. If you completed the tutorial in the last chapter, you've already created a few. Here, you'll add a background color to the page.

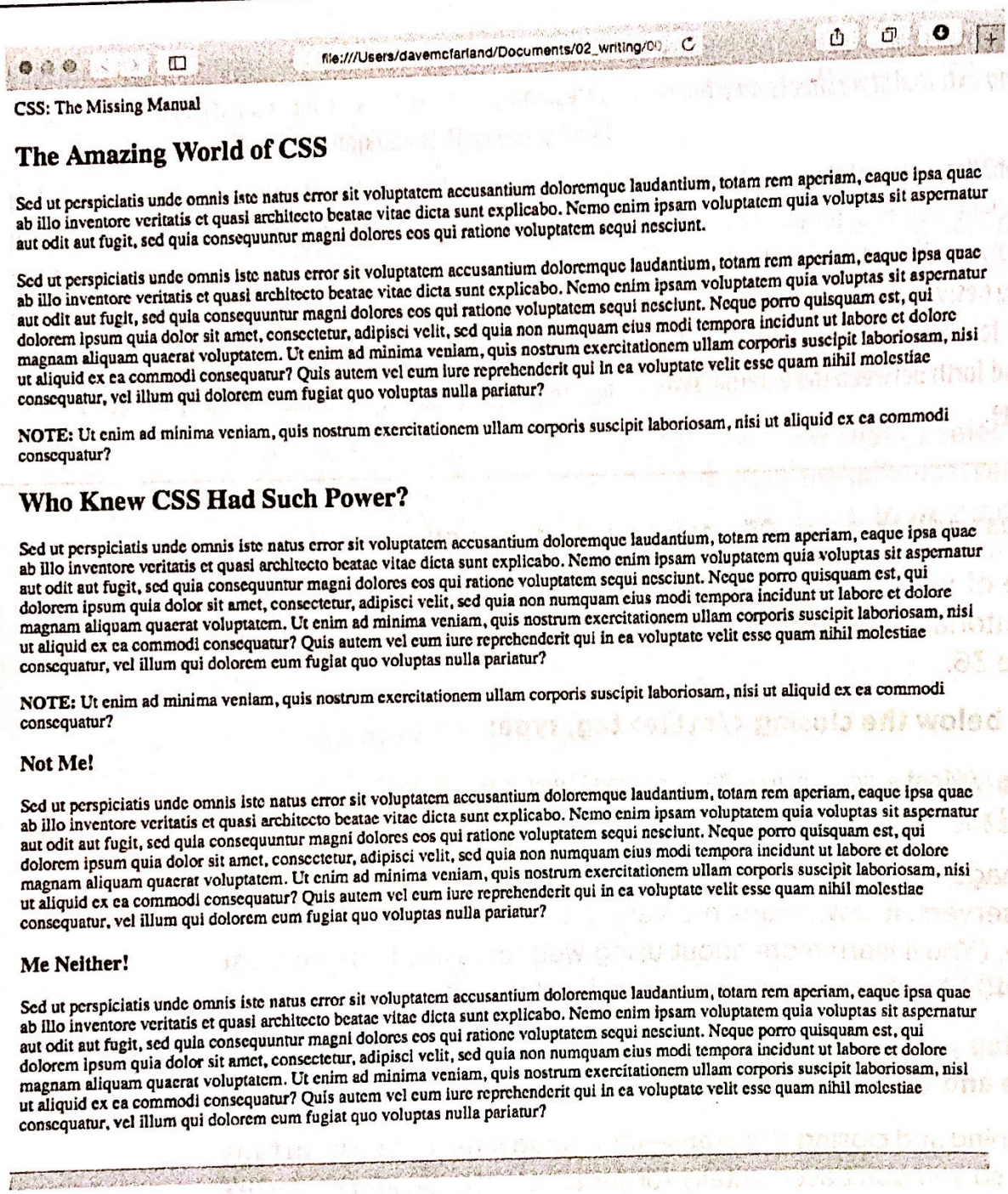

FIGURE 3-10
Plain HTML looks cold and monotonous in a web browser. But with a little CSS, you can turn drab (shown here) into fab (Figure 3-11) in 31 easy steps.

4. **Click between the opening and closing style tags you just added and type** body { **, hit Enter twice, and type the closing** }**.**

It's a good idea to always add the closing brace immediately after typing the opening brace, just so you don't forget. To create a tag selector, simply use the name of the HTML tag you wish to format. This style applies to the <body> tag. Now you can set the background color and the margin of space around the page.

5. **Click between <body> style's opening and closing braces ({ }) and add the following three CSS properties to supply the style's formatting—color, size, font, and left indent (as set by the padding):**

```
body {
  background-color: rgb(50,122,167);
  padding: 0 20px 20px 20px;
  margin: 0;
}
```

Press Enter to place each CSS property on its own line. It's also a good idea to visually organize your CSS code by indenting each property with the Tab key (some designers use two spaces instead of a tab—it's up to you).

The properties here change the background color of the page: `rgb()` is one way to specify a color's red, green, and blue values. In this case, the color is a dark blue. This background makes the black text hard to read, so you need to change the color of the paragraph tags.

NOTE These property names and their values may look unfamiliar. For now, just type them as is so you can get a taste of padding and margins work. You'll learn much more about these properties in Chapter 6.

6. **Add another style below the body style you just created:**

```
p {
    color: rgba(255,255,255,.6);
    font-size: 1em;
    font-family: "Varela Round", Arial, Helvetica, sans-serif;
}
```

With four CSS properties, this style supplies formatting for all paragraphs (all <p> tags)—color, size, and font. This time, the color is determined by an `rgba()` color value. With an extra "a" after the usual "rgb," this versatile value lets you create a color that's partially transparent. In this case, you've set the paragraph's text to white (which is 255,255,255 in rgb), but only 60 percent opaque (that's the .6 part). A bit of the blue background color shows through, so the paragraph text appears light blue.

Time for a look-see.

7. **Open the page in a web browser to preview your work.**

Unless you tinker with the preference settings, most browsers display black text in a standard serif font like Times. If your CSS style works properly, then you should see seven paragraphs using the Varela Round font in a light blue color.

Creating a Group Selector

Sometimes you'll want several different elements on a page to share the same look. For instance, you may want all your headings to have the same font and color for a

consistent style. Instead of creating separate styles and duplicating the same property settings for each tag—<h1>, <h2>, and so on—you can group the tags together into a single selector.

1. **Return to your text editor and the *selector_basics.html* file.**

   Let's add a new style below the <p> tag style you just created.

2. **Click at the end of the closing brace of the p tag selector, press Enter to start a new line, and add the following code:**

```
h1, h2, h3 {

}
```

   As explained earlier in this chapter, a group selector is simply a list of selectors separated by commas. This rule applies the same formatting, which you'll add next, to all <h1>, <h2>, and <h3> tags on the page.

3. **Click in the empty line between the opening { and closing } and add five CSS properties:**

```
color: rgb(255,255,255);
font-family: Arial, "Palatino Linotype", Times, serif;
border-bottom: 2px solid rgb(87,185,178);
padding-top: 10px;
padding-bottom: 5px;
```

   There's a lot going on here, but basically you're setting the color and font type for the headlines, adding a border line below the headlines for visual interest, and controlling the top and bottom spacing by using the padding property. The padding property adds space from the edges of an element without affecting a background or border—you're adding a bit of space above the headline, and inserting a bit of space between the bottom of that text and the border line below it.

4. **Save the file, and preview it in a web browser.**

   The <h1> heading near the top of the page and the <h2> and <h3> headings lower on the page all have the same font and font color as well as a greenish-blue border below them (see Figure 3-11). The <h1> tag looks a bit small, but you can easily bump up its size.

5. **Go back to your text editor and the *selector_basics.html* file. Add another style below the group selector style you just created:**

```
h1 {
  font-size: 2em;
}
```

   This style increases the size of the font. An em is the default browser font-size, so 2em is twice the normal text size. Notice, as well, that it's possible to have

more than one style apply to an element at the same time—the `h1, h2, h3` rule and the `h1` rule in this case. Both the group selector and this new type selector apply to `h1` tags on this page. This process is the *cascade* in Cascading Style Sheets. You'll learn all about how styles interact in Chapter 5.

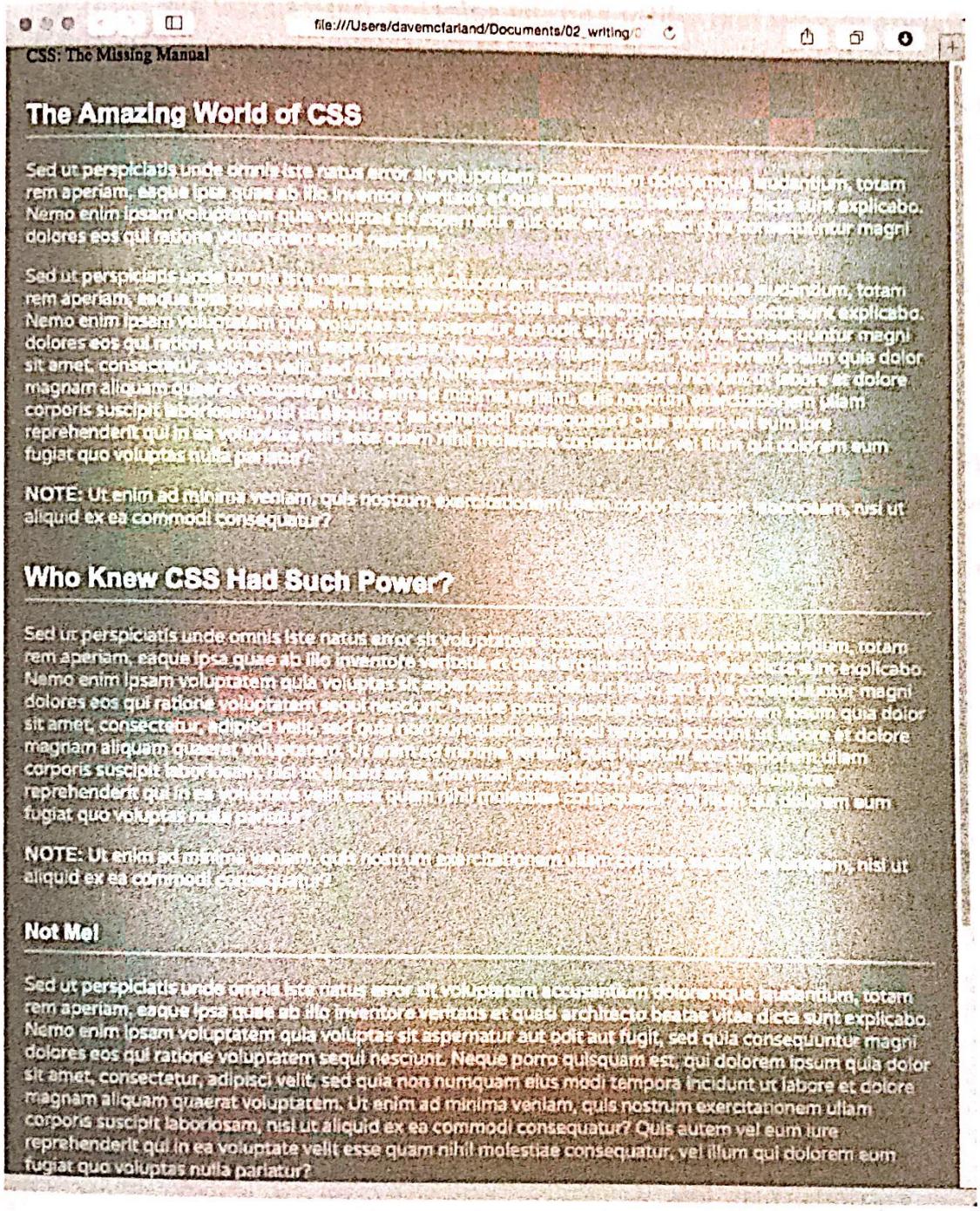

FIGURE 3-11
A simple type selector can completely transform the appearance of every instance of a tag, making quick work of styling all the paragraphs of text on a page. And in this case, a group selector does even more by formatting every instance of three different headline tags!

Creating and Applying an ID Selector

ID selectors are used to style a single tag. You create the style and add an ID attribute to a tag on the page, and then the properties from that style are applied to that single tag. You'll frequently use IDs to identify form elements, to create in-page links in a page (see the box on page 48), and to use JavaScript to control elements on a page.

Although many web designers now shy away from ID selectors (you'll learn exactly why on page 108), it's still good to know how to use them.

In this exercise, you'll create a style that controls the look of the text that appears at the very top of the page: "CSS: The Missing Manual." This text is considered the page's logo, and you'll create a special ID style to format it.

1. **Return to your text editor and the *selector_basics.html* file.**

   You'll add a new style below the h1 class style you created before.

2. **Click after the previous style's closing bracket (}), hit Enter to create a new line, and then type** `#logo {`.

   ID selectors always begin with the pound symbol (#). The style's name indicates that you'll apply this to a page element that's considered the site's logo.

3. **Hit Enter again, and then type:**

```
font-family: Baskerville, Palatino, sans-serif;
font-size: 2em;
color: rgba(255,255,255,.8);
font-style: italic;
text-align: center;
margin-bottom: 30px;
background-color: rgb(191,91,116);
border-radius: 0 0 10px 10px;
padding: 10px;
```

   It looks like a long list of properties, but all it does it set some font properties, background color, and spacing for the logo text.

4. **Finish the style by typing the closing brace. The whole thing should look like this:**

```
#logo {
  font-family: Baskerville, Palatino, sans-serif;
  font-size: 2em;
  color: rgba(255,255,255,.8);
  font-style: italic;
  text-align: center;
  margin-bottom: 30px;
  background-color: rgb(191,91,116);
  border-radius: 0 0 10px 10px;
  padding: 10px;
}
```

   If you save the file and preview it in a web browser, you won't see any difference. That's because this style doesn't *do* anything until you apply it. So you'll add an ID attribute to the page's HTML, indicating where you want the ID style to apply.

5. **Find the tag near the opening <body> tag—it has the text "CSS: The Missing Manual" in it. Add id="logo" to the opening div so the HTML looks like this:**

```
<div id="logo">
  CSS: The Missing Manual
</div>
```

The tag now reflects the formatting defined in the #logo style. As with all things CSS, there are many ways to arrive at the same destination: You could instead use a class style and apply it to the tag. But in this case you're using an ID selector, since the point of this style—identifying the logo on the page—is in keeping with the general notion of ID selectors.

6. **Save the page, and preview it in a browser.**

Now the "CSS: The Missing Manual" text is centered, light-colored, and inside a small box at the top of the page (Figure 3-12).

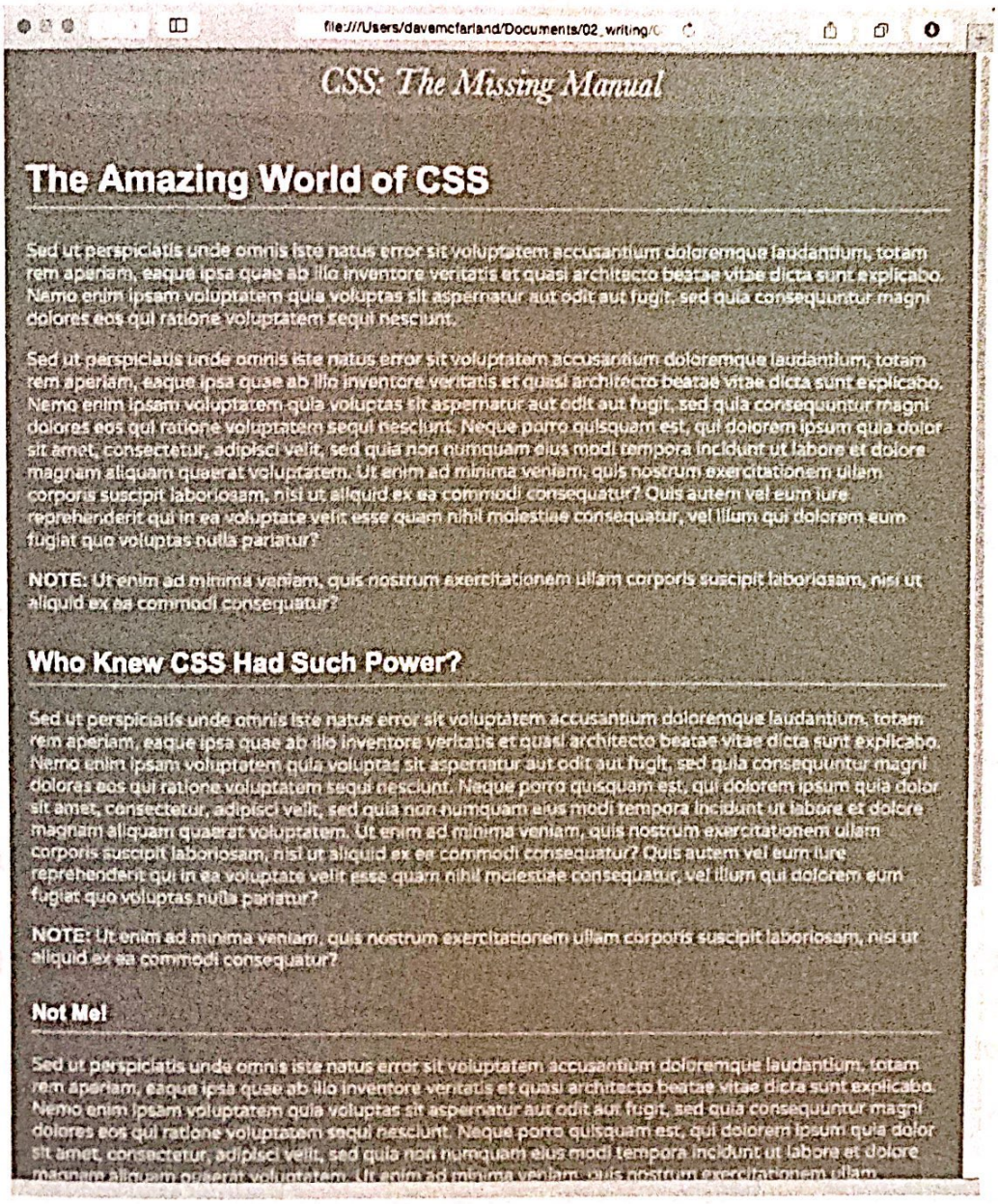

FIGURE 3-12
ID selectors are one way to style unique elements, like the logo bar at the top of this page.

Creating and Applying a Class Selector

Type selectors are quick and efficient, but they're a bit indiscriminate in how they style a page. What if you want to style a single <p> tag differently than all the other <p> tags on a page? A class selector is the answer.

1. **Return to your text editor and the *selector_basics.html* file.**

   Add a new style below the ID selector style you just created.

2. **Click at the end of the closing brace of the `#logo` selector, press Enter, and then type**

   ```
   .note {

   }
   ```

   This style's name, `.note`, indicates its purpose: to highlight paragraphs that contain extra bits of information for your site's visitors. Once you create a class style, you can apply it wherever these notes appear—like the third paragraph in this page.

3. **Click in the empty line between the opening { and closing } and add the following list of properties to the style:**

   ```
   color: black;
   border: 2px solid white;
   background-color: rgb(69,189,102);
   margin-top: 25px;
   margin-bottom: 35px;
   padding: 20px;
   ```

   Notice that you're not using the `rgb()` color values to color the font and border. CSS has several different ways to specify a color, including keywords like `white`, `black`, or `orange`. You'll learn about those on page 147.

   If you preview the page now, you see no changes. Like ID selectors, class selectors don't have any effect on a web page until you apply the style in the HTML code.

4. **In the page's HTML, there are two `<p>` tags that begin with the word "Note" inside `<strong>` tags.**

   To apply a class style to a tag, simply add a class attribute, followed by the class selector's name—in this case, the note style you just created.

5. **Click just after the `p` in the first `<p>` tag, and then type a space followed by `class="note"`. The HTML should now look like this (what you just typed is in bold):**

   ```
   <p class="note"><strong>NOTE:</strong>
   ```

Be sure *not* to type `class=".note"`. In CSS, the period is necessary to indicate a class style name; in HTML, it's verboten. Repeat this step for the second paragraph (it's just above the `<h3>` tag with the text "Not Me!").

NOTE There's no reason you can't add this class to other tags as well, not just the `<p>` tag. If you happen to want to apply this formatting to an `<h2>` tag, for example, then your HTML would look like this:

```
<h2 class="note">
```

6. **Save and preview the web page in a browser.**

   The note paragraph is nicely highlighted on the page (see Figure 3-13).

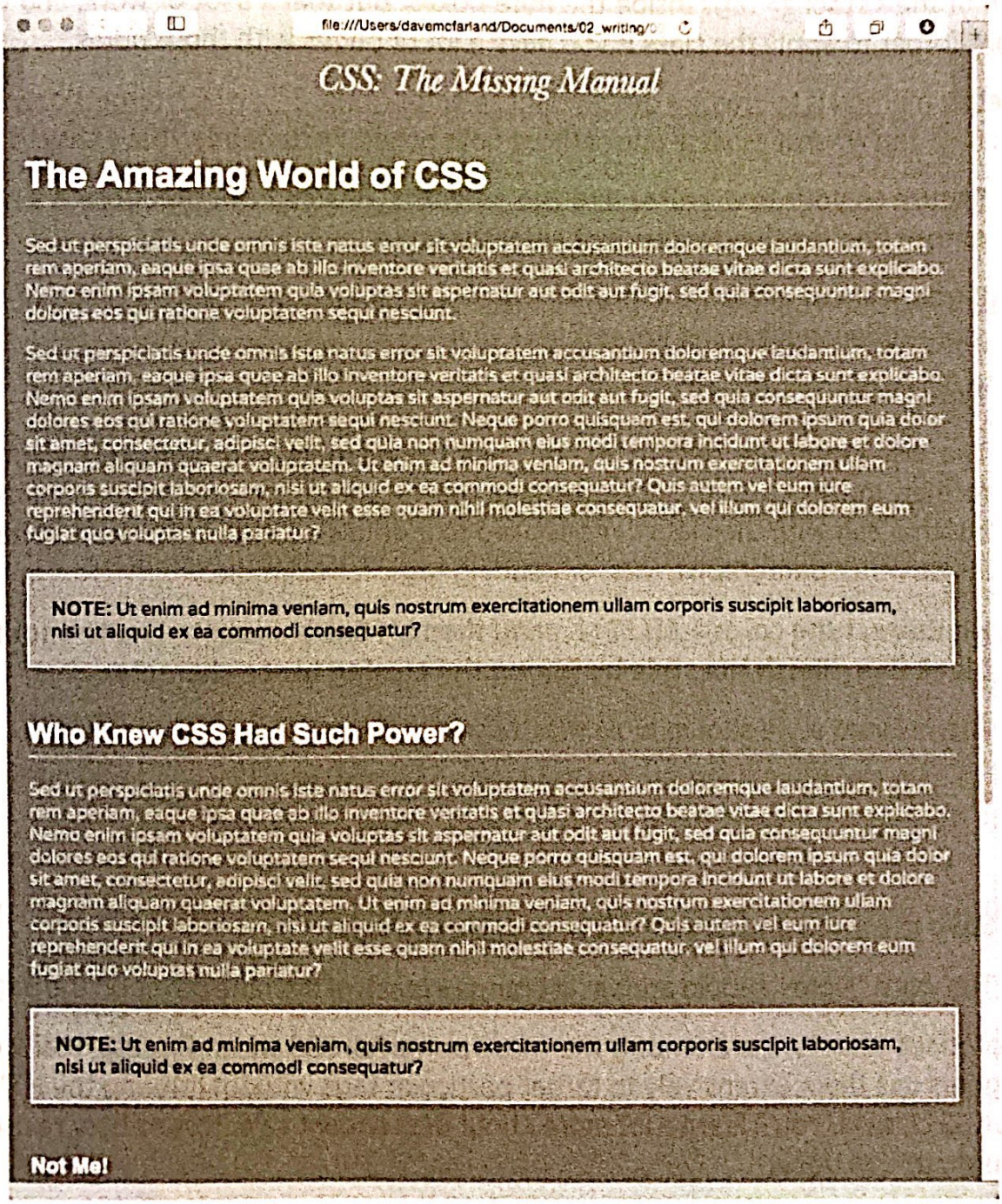

FIGURE 3-13
You can make detailed formatting changes with class selectors. A class style gives selected paragraphs different formatting from all other paragraphs on the page. The distinctive note box pictured here uses a class style to stand out from the crowd.

NOTE If your page doesn't look like Figure 3-13, then you may have mistyped the name of a property or its value. Double-check your code with the steps above. Also, make sure to end each declaration (`property: value` combination) with a semicolon and conclude the style with a closing brace at the very end. When your style is not working correctly, missing semicolons and closing braces are frequent culprits.

Creating a Descendant Selector

On the *selectors_basics.html* page, you applied the note class to two paragraphs. Each of those paragraphs begins with the word "Note:" in bold—actually the word is wrapped inside the HTML `<strong>` tag, which all browsers display as bolded text. But what if you want to format those bolded words in bright orange? You could create a tag style for the `<strong>` tag, but that would affect all `<strong>` tags on the page, and you only want to change the strong tag inside those note boxes. One solution would be to create a class style—`.noteText`, for example—and apply it to each of the `<strong>` tags inside the note boxes. But that's a lot of work, and you might forget to apply the class if you have a lot of pages with those notes.

A better method is to create a descendant selector (page 50), which targets only the `<strong>` tag when it's inside one of these note boxes. Fortunately, that's easy to do.

1. **Return to your text editor and the *selector_basics.html* file. Create a new empty line for the descendant selector style.**

   If you just completed the previous steps, click after the closing brace of the *.note* style, and then hit Enter.

2. **Type `.note strong {`.**

   The last tag in the selector—`strong`—is the element you ultimately want to format. In this case, the style formats the `<strong>` tag only when it's *inside* another tag with the class `note` applied to it. It has no effect on `<strong>` tags inside other paragraphs, lists, or heading tags, for example.

3. **Hit Enter, type `color: #FC6512;`, and then hit Enter again to create another blank line. Finish the style by typing the closing brace character.**

   The finished style should look like this:

   ```
   .note strong {
     color: #FC6512;
   }
   ```

4. **Save the page and preview it in a web browser.**

   The word "Note:" should appear in orange in each of the note boxes on the page.

Descendant selectors are among the most powerful CSS tools. Professional web designers use them extensively to target particular tags without littering the HTML with CSS classes. You'll learn a lot more about descendant selectors as they're used throughout this book.

Finishing Touches

The text on this page expands to fill the browser window. To see this in effect, preview the page and resize your browser window. You'll see the lines of text get wider as you stretch the window. If you have a large monitor, you'll see that the text becomes pretty hard to read past a certain width: The lines of text are too long to read comfortably. Fortunately, you can set a width for the page's content, so it doesn't get too wide to read.

1. **Return to your text editor and the *selector_basics.html* file. Create a new empty line for a new style.**

   If you just completed the previous steps, click after the closing brace of the `.note strong` descendant selector style, and then hit Enter.

2. **Add another style:**

```
article {
  max-width: 760px;
}
```

   This is another type selector. It applies to the HTML5 `<article>` tag, which is used to define the content that makes up an article like a blog post, or the content on this page.

   The `max-width` property sets the maximum width for the tag, meaning that the article tag will never get wider than 760 pixels. Save the file and preview it in a browser. If you widen the browser window, you'll notice that past 760 pixels, the window gets wider and the blue background of the page appears, but the text no longer expands.

   On the other hand, if you make the browser window smaller than 760 pixels, the text lines do get shorter. That's the power of the `max-width` property, which sets a maximum width, but no minimum. This property is extremely useful when you're designing sites that need to work on a variety of screen sizes—desktop computers, laptops, tablets, and smart phones. It's an important part of *responsive design*, which you'll learn about in Chapter 17.

   Now that you've limited how far the text can expand, it would be nice to keep the content centered on the screen instead of sticking to the left edge as the browser window expands.

3. **Add one more property to the `article` style so it looks like this (addition in bold):**

```
article {
  max-width: 760px;
  margin: 0 auto;
}
```

The `margin` property sets the space between an element and other elements around it. You'll read more about margins on page 187, but here this line sets the left and right margins to auto, which tells the web browser to automatically figure out how much space to add to the left and right sides of the article tag. Once the browser window gets past 760 pixels, the article element gets no wider, so the browser basically adds empty space to both the left and right of the tag, in essence centering it in the middle of the browser window.

For fun, you'll add one more advanced style—an adjacent sibling selector discussed on page 67—to format the paragraph immediately following the first headline on the page. (You can achieve the same effect by creating a class style and applying it to that paragraph, but the adjacent sibling selector requires no changes to your HTML.)

4. **Add one last style:**

```
h1+p {
 color: rgb(255,255,255);
 font-size: 1.2em;
 line-height: 140%;
}
```

This style will apply to any paragraph that *immediately* follows an <h1> tag—in other words, the first paragraph after the top headline on the page. It won't apply to the second or any subsequent paragraphs. This selector provides an easy way to create a unique look for an introductory paragraph to set it off visually and highlight the beginning of an article.

The style changes the font color and size. The `line-height` property (which you'll read about on page 163) controls the space between lines in a paragraph (also known as *leading*).

If you preview the page now, you'll see that the top paragraph is white and its text is larger, and there's more space between each line of text (see Figure 3-14). If you actually deleted this paragraph in the HTML, you'd see that the remaining paragraph would suddenly be white with larger text, since it would be the new adjacent sibling of the <h1> tag.

And there you have it: a quick tour through various selector types. You'll get familiar with all of these selectors (and more) as you go through the tutorials later in the book, but by now, you should be getting the hang of the different types and why you'd use one over the other.

NOTE You can see a completed version of the page you've just created in the *03_finished* folder.

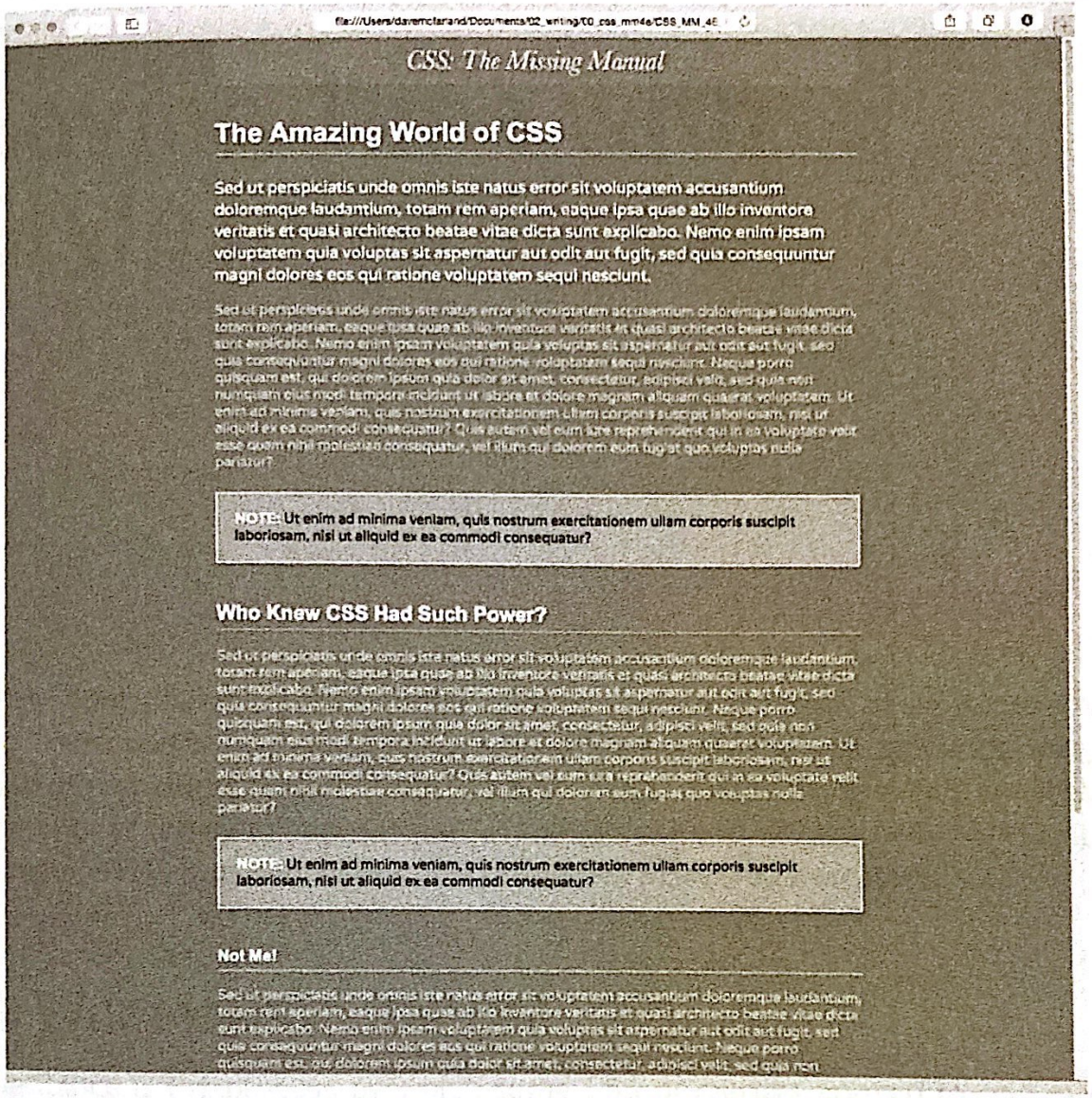

FIGURE 3-14

The page has really come together. The set width and the typographic details have really improved the look of the boring HTML page you started with.