CHAPTER 8

CSS 101

```
<table border="1" width="480" c
<tr><td width="120"><font face=
color="gray"><b>The Legend</b><

<tr><td><font face="Arial, Helv
is the name of a </font><i><fon
color="black">fictional giant a
Geneva" size="3" color="black">
Helvetica, Geneva" size="3" col
face="Arial, Helvetica, Geneva"
several works since 1933. Most
groundbreaking 1933 film, the f
sequels.</font>
<br>&nbsp;<br>
<font face="Arial, Helvetica, G
film, the character's name is K
```

FAMOUS PR

King Kong

The Legend

King Kong is the name of a *fictional g*
several works since 1933. Most of thes

FAMOUS PR

King Kong

The Legend

King Kong is the name of a *fictional g*
several works since 1933. Most of the
film remakes of 1976 and 2005, and r

We've spent the first half of the book building a solid foundation, teaching you to create markup using well-structured XHTML. Now it's time to add some style to your well-crafted web pages using CSS.

The impatient among you have been waiting for this chapter for some time or have possibly cheated and skipped a few chapters, perhaps because you mistakenly think this chapter marks the beginning of the "design" chapters. If you can recognize yourself in this description, particularly if you skipped the first half of the book because you wanted to dive straight into CSS, please—we urge you—skip back a few chapters.

Why?

Simple. We've been covering design for some time now. All that *structured markup*, all that talk of *semantics*, all that "choose the right tag for the job"—all of that *is* design.

Good design *isn't* choosing this year's color or selecting a typeface that's hot. Good design is much more fundamental than that; it's about taking some information and giving it structure. It's about amplifying meaning by drawing out an information hierarchy and teasing out semantics. In short, it's . . . everything we covered in the first half of the book.

If you're feeling a little guilty—you skipped a few sections here and there, you really wanted to get to "the exciting parts"—**we strongly recommend you skip back** and read the preceding chapters. You'll thank us if you do.

Not guilty? Read on!

This chapter introduces the fundamentals of Cascading Style Sheets (CSS), the presentation part of the *content plus presentation* equation. You'll learn the basic principles of CSS and start applying style to some of the web pages you've created in the first half of the book.

CSS is a language used to control the presentation of documents written in markup language, for example, to style web pages written in XHTML. CSS can be used to define colors, fonts, and a variety of other aspects of document presentation; it can also be used to position elements and control layout.

CSS is primarily designed to enable the separation of a document's content (written in XHTML) from a document's presentation (written in CSS). This separation can improve content accessibility, provide users with more flexibility in accessing content, and allow the same markup to be presented in a variety of different styles for different rendering methods, for example, for viewing on screen and in print or for accessing via a screen reader.

Up until this point our well-structured web pages have been styled using the browser's default style sheet. Although these style sheets vary slightly between browsers, they have common characteristics including black text, blue text for links, and purple text for visited links. By adding your own style sheet, you can override the browser's style sheet, allowing you to style the page as you wish.

By the end of this chapter you'll be ready to start adding additional—visual and presentational—design to your web pages, turning them from well-structured web pages into well-structured *and* well-designed web pages. Good times.

Adding some style

You should by now be capable of building well-structured web pages using semantic markup. You should also know how to upload these to some personal web space as covered in Chapter 7. This is a significant achievement. However, as things stand your web pages aren't too far removed from what the Web was supposed to look like circa 1993. You might find it hard to believe, but that is *not* a bad start.

Now you'd really like to work on the presentation aspect, controlling the look and feel of your web pages to enhance their design. This chapter takes your well-structured web pages and begins to apply some style to them using CSS. Before we embark on this part of our journey, it's worth a quick refresher on what HTML was and, equally importantly, *wasn't* intended for.

HTML: A brief refresher

As we discussed in Chapter 1, when HTML was invented, it was not intended to be a presentational language. The tags that dealt with the visual aspect of text (the choice of typeface, its size and color, for example) were limited and were intended only to introduce a semantic structure to text. We covered this in Chapter 3 when we discussed structured markup, where we used an `<h1>` to signify "I am more important than an `<h2>`" and so on.

As web browsers grew more sophisticated, adding support for images and introducing additional—often proprietary—tags, web designers grew more and more adventurous and started to embrace non-semantic markup to create hidden scaffolding for increasingly complicated designs.

The introduction of WYSIWYG (What You See Is What You Get) editors that allowed the user to create HTML by dragging and dropping visual elements on the page, much like in a word processor, created markup of such complexity that it would be almost impossible to create by hand, or to change or edit without access to the program that created the markup in the first place.

Web pages created using WYSIWYG editors (for example, FrontPage or Dreamweaver) took longer to download due to the amount of extra markup they contained, they were harder to update and maintain, and if you wanted to change a visual element on your web site, you had to go through each and every individual page, as the visual style was embedded in the content of the page and not in a separate design file or style sheet.

Working with pages generated in WYSIWYG editors? In a word: nightmare.

CSS offered the possibility to change all that, separating content and presentation, which forms the cornerstone of the Web Standardistas' approach.

CSS isn't new

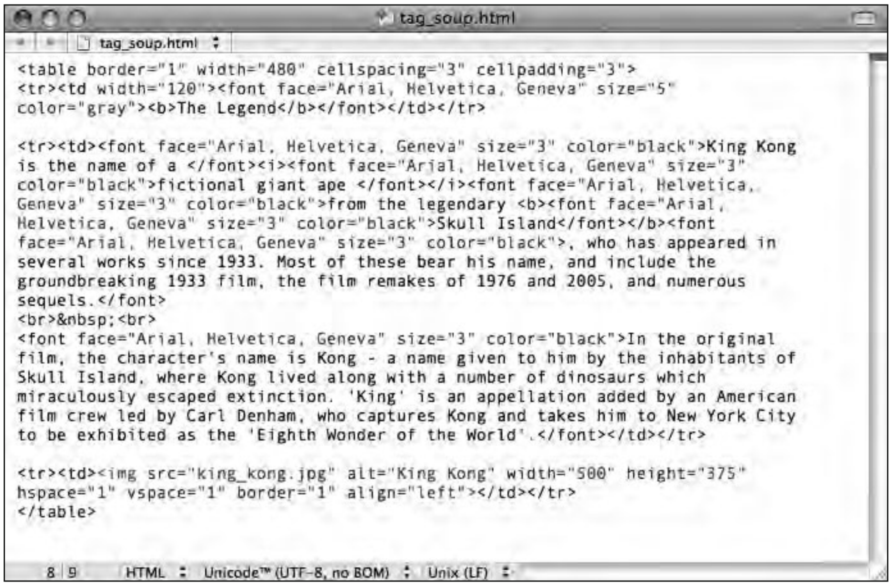
CSS has been around for over a decade, being proposed as early as 1994 and agreed upon as a W3C Recommendation in December 1996, but it is only in the last few years that browser support has been reliable enough for web designers to depend on CSS for layout and design.

The key advantages of CSS include the fact that it's easy to maintain; it helps in the creation of lean web pages (look at the tag soup example in the next section to understand why this is a good thing); it uses the same XHTML markup for screen and print; and it can be used to improve accessibility, for example, through the inclusion of a high-contrast style sheet for visually impaired users.

Tag soup or lean and mean?

We introduced the concept of *tag soup* in Chapter 1; we'll now show you an example we've reverse-engineered to demonstrate how much smaller a well-formatted web page can become by removing presentational HTML.

In Figure 8-1, we've marked up a section of our King Kong page using `<font>` tags to style the text and `<table>` tags to control the layout, as would have been typical before CSS was widely supported by standards-compliant browsers.



```
<table border="1" width="480" cellspacing="3" cellpadding="3">
<tr><td width="120"><font face="Arial, Helvetica, Geneva" size="5"
color="gray"><b>The Legend</b></font></td></tr>

<tr><td><font face="Arial, Helvetica, Geneva" size="3" color="black">King Kong
is the name of a </font><i><font face="Arial, Helvetica, Geneva" size="3"
color="black">fictional giant ape </font></i><font face="Arial, Helvetica,
Geneva" size="3" color="black">from the legendary <b><font face="Arial,
Helvetica, Geneva" size="3" color="black">Skull Island</font></b><font
face="Arial, Helvetica, Geneva" size="3" color="black">, who has appeared in
several works since 1933. Most of these bear his name, and include the
groundbreaking 1933 film, the film remakes of 1976 and 2005, and numerous
sequels.</font>
<br>&nbsp;<br>
<font face="Arial, Helvetica, Geneva" size="3" color="black">In the original
film, the character's name is Kong - a name given to him by the inhabitants of
Skull Island, where Kong lived along with a number of dinosaurs which
miraculously escaped extinction. 'King' is an appellation added by an American
film crew led by Carl Denham, who captures Kong and takes him to New York City
to be exhibited as the 'Eighth Wonder of the World'.</font></td></tr>

<tr><td><img src="king_kong.jpg" alt="King Kong" width="500" height="375"
hspace="1" vspace="1" border="1" align="left"></td></tr>
</table>
```

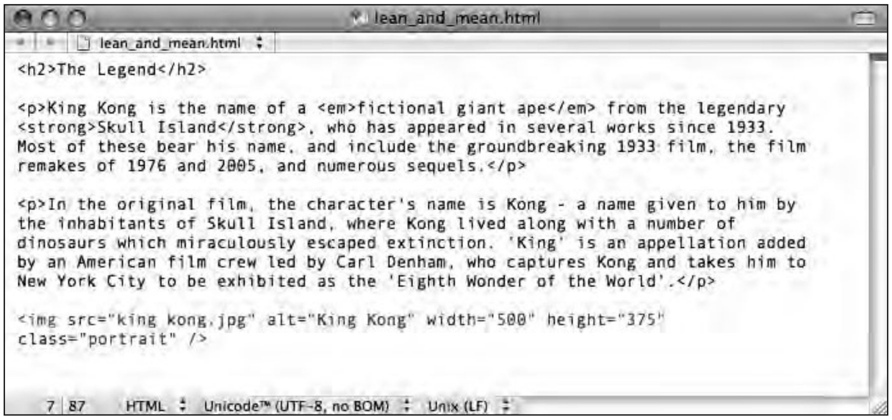
Figure 8-1. Our reverse-engineered tag soup example is not only unwieldy, but also uses tables for layout, a Web Standardista faux pas.

Take a look at the paragraphs in Figure 8-1. As you can see, they haven't even been marked up with `<p>` tags, relying instead on a considerable amount of presentational markup. Every single time a paragraph is encountered, it's been styled with the addition of `<font face="Arial, Helvetica, Geneva" size="3" color="black">`, setting the paragraphs to display in Arial, size 3, and black (we'll explain why the additional typefaces in the `<font>` tag are included in Chapter 9). Note how this is added to *every single paragraph!*

As you can see, this could get really messy and hard to read, not to mention hard to write. As if that weren't bad enough, this complex and nonstructured markup is much harder for search engines to index.

Surely there must be a more efficient way to do this. Imagine we have a client who would prefer Times New Roman to Arial and would rather the paragraphs were in red, not black. The client would also like the text size increased just a little. We would have to find *every instance* of `<font face="Arial, Helvetica, Geneva" size="3" color="black">` and replace it with `<font face="Times New Roman, Times, Georgia" size="4" color="red">`. No small task, especially if we had a web site with hundreds of pages.

In the Figure 8-2, we take the same page and remove the presentational `<font>` and `<table>` tags, reducing the markup considerably.



```
<h2>The Legend</h2>

<p>King Kong is the name of a <em>fictional giant ape</em> from the legendary
<strong>Skull Island</strong>, who has appeared in several works since 1933.
Most of these bear his name, and include the groundbreaking 1933 film, the film
remakes of 1976 and 2005, and numerous sequels.</p>

<p>In the original film, the character's name is Kong - a name given to him by
the inhabitants of Skull Island, where Kong lived along with a number of
dinosaurs which miraculously escaped extinction. 'King' is an appellation added
by an American film crew led by Carl Denham, who captures Kong and takes him to
New York City to be exhibited as the 'Eighth Wonder of the World'.</p>

<img src="king kong.jpg" alt="King Kong" width="500" height="375"
class="portrait" />
```

Figure 8-2. Lean and mean, our web standards–based web page is not only considerably smaller, but far more accessible. It's also more search engine friendly.

These contrasting examples go to the very heart of the problem. The use of `<font>` tags and other presentational markup is inefficient and results in much larger file sizes, which take longer to download and longer to edit and update. The fragment in Figure 8-1 is 1,396 characters long, while the fragment in Figure 8-2 is only 779 characters, *almost half the size*. It will download twice as fast, and we can update its style and presentation considerably more quickly.

So how do we add all the presentational information that's not included in the second example? The answer is using CSS.

CSS to the rescue

Using CSS to add style to your web pages removes the need for all the individual `<font>` tags encountered in the previous section. By providing a mechanism for selecting specific elements within your markup—for example, all the `<p>` tags in the preceding example—CSS allows you to target them and style them as you wish. Write one CSS rule and style *every* `<p>`. This results in web pages that are efficient, much easier to maintain, and faster to download.

While early web authors and designers had little choice but to employ elements in a presentational manner to create visually effective web pages, CSS has largely done away with the need for those methods. Old habits die hard, however, and many authors continue to unnecessarily employ these tag soup hacks rather than embrace CSS. As a Web Standardista, you know better.

Using CSS to control presentation—adding a layer of style to your well-structured XHTML pages—allows you to separate content and presentation. Focus first on the creation of well-structured markup, markup that uses the right tag for the job, before styling that markup and controlling how it displays using CSS. This is a much better approach, and one wonders why it's taken so long for CSS to take a hold.

This all sounds great, but how do we actually use CSS?

Meet CSS

Although XHTML and CSS are two different languages, with different rules, CSS is easy to learn and, once you understand the basic principles of the language, you should be up and running in no time.

Let's take a look at a simple CSS rule. We'll work with a typical piece of markup from the bad old days and demonstrate how you can use CSS to replace the cumbersome `<font>` tags used. In the tag soup example in Figure 8-1, we showed how `<font>` tags had been used to style the different paragraphs on the King Kong page. We've simplified the markup here to style just one aspect of the type, its color:

```
<font color="teal">King Kong is the name of a fictional giant ape
 from the legendary Skull Island...</font>
```

In the tag soup days, when we wanted to style a paragraph, we needed to include the additional markup in the `<font>` tag, *every single time*. With CSS used to style a solid foundation built using structured markup, we can achieve this much more efficiently.

The first step in the process, as you know by now, is to remove the `<font>` tags and wrap this admittedly short paragraph in `<p>` tags as in the following example:

```
<p>King Kong is the name of a fictional giant ape from the legendary
Skull Island...</p>
```

Not only is the inclusion of `<p>` tags semantic, a good thing as you know, but it also gives us an element we can target with CSS. The result of this is that it achieves exactly the same effect as multiple `<font>` tags, styling *all* of our `<p>` elements, using just the following CSS rule:

```
p
{
color: teal;
}
```

Although this might seem a little complicated at first, don't worry, all will soon be revealed as we break down this basic CSS rule in the next section. The preceding CSS rule does exactly the same as the `<font>` tags in the first example, but as we can write one rule to target all instances of `<p>` on a page, it is a great deal more efficient.

Remember the client who preferred the color red? Simply change the preceding rule as follows, and it will change all occurrences of `<p>`, styling them red, and the client will be happy:

```
p
{
color: red;
}
```

We'll now introduce you to how the preceding CSS rule is structured, breaking it down into its constituent parts.

Anatomy of a CSS rule

At first glance, the rule in the previous example probably appears a little bit cryptic; however, once we've explained it, you'll soon understand what's going on.

CSS rules are comprised of a **selector**, a **property**, and a **value** as follows:

```
selector
{
property: value;
}
```

In Figure 8-3, we've taken this example and rewritten it on one line to illustrate its different components more clearly.

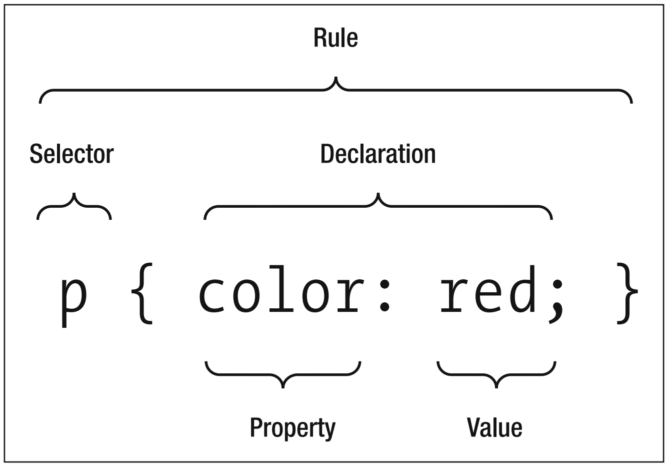


Figure 8-3. The different parts of a very simple CSS rule, styling just one property

Let's look again at this simple rule and break it down into its key components:

```
p
{
color: red;
}
```

- The letter `p` on the first line targets the `p` element (i.e., all content within `<p>` tags on the page).
- Everything between the { and } (curly brackets) is style we're applying to the `p` elements.
- The declaration `color: red;` specifies a particular color for those elements.

In CSS a **declaration** is the name given to a `property: value;` pair. In the preceding example, the `property` we're targeting is the color, and its `value` is being set to `red`. As you'll see in the section "A slightly more complex rule," we can identify a number of `property: value;` pairs to create more complex rules. The property and value are separated with a colon (`:`).

It's also worth noting the semicolon at the end of the declaration line. We use semicolons to mark the end of each declaration; we'll need them when we start adding additional declarations to our rules as in the examples in the following section. The semicolon is important and needs to be included; the semicolon, *not* the line break, tells the interpreter that this is the end of the declaration.

A note on formatting

Before we go on it's worth pausing for a moment to talk about how CSS rules can be formatted. If you've been using View Source to take a look at how others' web pages are constructed, you might see a number of variations on how CSS rules are laid out. It's worth noting that the following three examples are all functionally equivalent:

```
p { color: red; }
```

or

```
p {
   color: red;
  }
```

or

```
p
{
color: red;
}
```

Based on our teaching experience, we find the last example to be the easiest layout for beginners to follow as it breaks the selector and property and value pairs onto separate lines. This makes the declaration easier to understand and follow.

A slightly more complex rule

So, now you know what a CSS rule looks like and we've introduced you to a simple rule to style all of our paragraphs red. However, you'd like to do a little more than change your paragraphs to red; you'd like to choose a typeface for them.

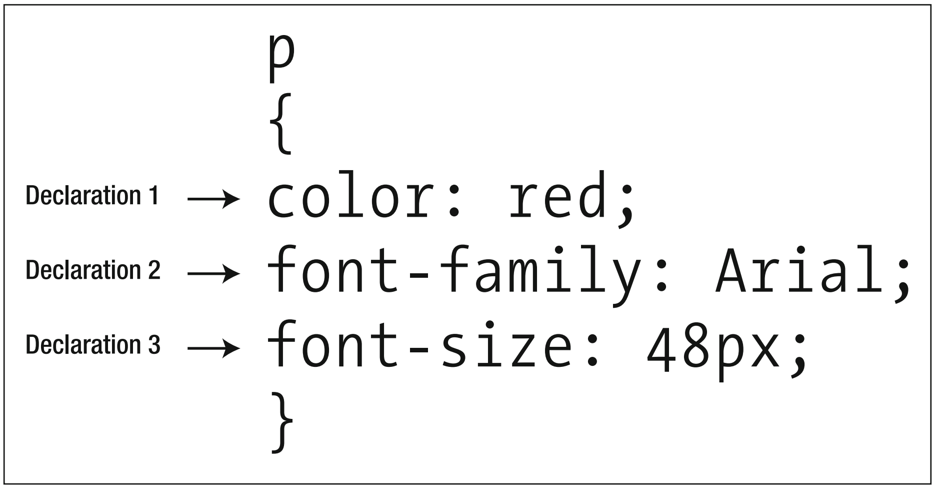
We could do this by writing a rule for each aspect we'd like to style as follows:

```
p
{
color: red;
}

p
{
font-family: Arial;
}
```

Although this will work, styling all the paragraphs in red and setting them in Arial, it would quickly become cumbersome, with endless rules styling each element. Good news, CSS allows us to combine declarations as in the following example, which does exactly the same thing:

```
p
{
color: red;
font-family: Arial;
}
```

Figure 8-4 shows an example of a CSS rule with multiple declarations, one per line.



```
p
{
color: red;
font-family: Arial;
font-size: 48px;
}
```

Figure 8-4. A more complex CSS rule, consisting of three declarations, each on its own line

We can add as many declarations as we want to each rule, enabling us to apply specific styles to each element on a page as in the following example:

```
p
{
font-family: Arial;
font-size: 12px;
font-weight: bold;
color: red;
background-color: yellow;
}
```

As you can see, although this example is getting quite complicated, it still follows the basic principles we established earlier. As your CSS rules become more complicated, you'll find them easier to read if you put each declaration on a separate line. Although not strictly necessary, this can help to establish exactly what a specific rule is targeting.

Now that you know what a CSS rule looks like, we need to show you how to add one to a web page. We'll introduce this in the next section, working on our King Kong page to add some basic style to it.

Adding CSS to a web page

Now that you've seen what CSS rules look like, we need to add them to our XHTML pages. There are two primary ways to do this: using either an embedded style sheet, where the style sheet is on the page itself, or an external style sheet, where the style sheet is an external file that is linked to.

While using an external style sheet—one CSS file that controls *all* the pages in our web site—will be our ultimate goal, we'll be focusing on embedded style sheets for the next few chapters.

When embarking on a project, it's often easiest to start with an embedded style sheet to work out basic issues like styling and layout. Having everything—XHTML and CSS—on one page makes developing and fine-tuning a little easier, as everything is located in one place, allowing you to test the effect of your CSS on your markup within a single file. Once you've reached a point at which you're happy with your CSS, it can be offloaded to an external style sheet which, when linked to by each page, will style all the pages in your web site. We'll cover external style sheets in Chapter 13.

One other method—which we'll just mention briefly—is the use of inline styles. Not too dissimilar to the `<font>` tag we introduced previously, CSS can be used inline to style elements as they occur, as in the following example:

```
<p style="color: red;">King Kong is the name of a fictional giant
ape from the legendary Skull Island...</p>
```

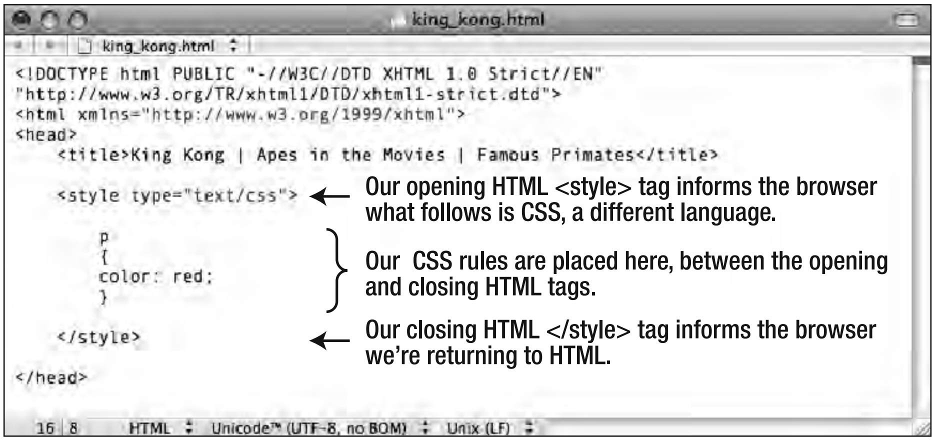
While inline styles can be useful in edge case scenarios, we've chosen not to cover them in this book, focusing instead on embedded and external style sheets.

Adding an embedded style sheet

When embedding a style sheet, we add our various CSS rules within a `style` element, which we place within the `head` as in the following example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>King Kong | Apes in the Movies | Famous Primates</title>
  <style type="text/css">

    p
    {
    color: red;
    }

  </style>
</head>
...
```

The `style` element is simply another XHTML element, which informs the browser that everything between the opening `<style...>` and closing `</style>` tags is CSS. The `type` attribute is required; it specifies the language of the elements contained within the `<style>` tags. Put simply, it informs the browser that we're breaking out of XHTML and entering into CSS (`text/css`). Figure 8-5 visualizes this.

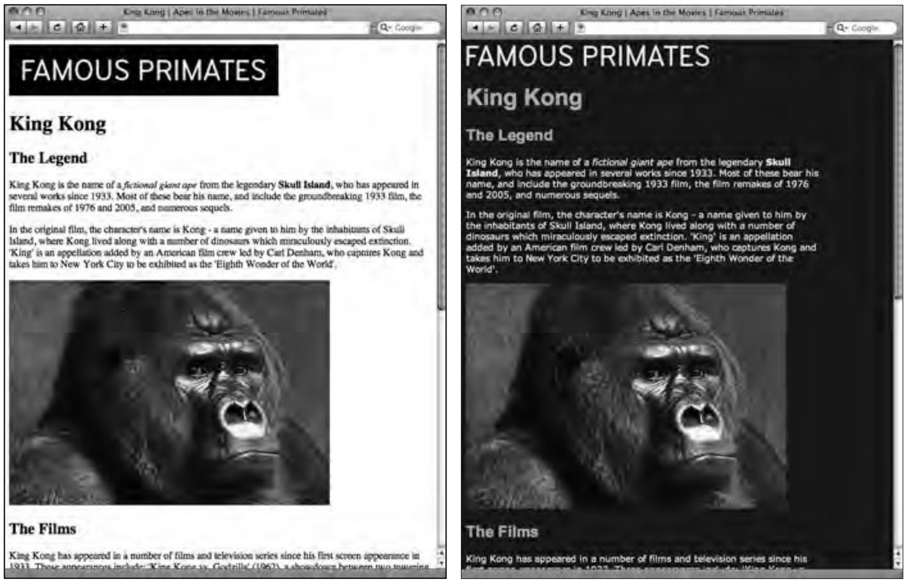


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>King Kong | Apes in the Movies | Famous Primates</title>

    <style type="text/css">

        p
        {
        color: red;
        }

    </style>

</head>
```

king_kong.html
Our opening HTML <style> tag informs the browser what follows is CSS, a different language.
Our CSS rules are placed here, between the opening and closing HTML tags.
Our closing HTML </style> tag informs the browser we're returning to HTML.
16 8 HTML Unicode™ (UTF-8, no BOM) Unix (LF)

Figure 8-5. Our simple CSS rule, located in an embedded style sheet in the head element

Let's take a look at an actual example—the King Kong page we last worked with in Chapter 6—and see how adding CSS to an unstyled XHTML page works in action. We compare this page unstyled and with some basic style added in Figure 8-6.



FAMOUS PRIMATES
King Kong
The Legend
King Kong is the name of a fictional giant ape from the legendary Skull Island, who has appeared in several works since 1933. Most of these bear his name, and include the groundbreaking 1933 film, the film remakes of 1976 and 2005, and numerous sequels.
In the original film, the character's name is Kong - a name given to him by the inhabitants of Skull Island, where Kong lived along with a number of dinosaurs which miraculously escaped extinction. 'King' is an appellation added by an American film crew led by Carl Denham, who captures Kong and takes him to New York City to be exhibited as the 'Eighth Wonder of the World'.
The Films
King Kong has appeared in a number of films and television series since his first screen appearance in

Figure 8-6. Our unstyled King Kong page as it stood at the end of Chapter 6 alongside the same page with some basic style added with CSS

A simple walkthrough

We'll start with the King Kong page that we've been working on in the previous chapters. As the page currently stands it has no style added and so displays using the browser's default style sheet. It's simply an unstyled, but well-structured XHTML page.

In this walkthrough we'll style the `body`, `h1`, `h2`, and `p` elements to introduce you to the fundamental principles of adding CSS to a web page, but before we get our hands dirty with the CSS, there's just one more thing we need to mention: colors.

Getting colorful

In the preceding examples, we've used `keywords` to specify our colors, for example, `red`, `teal`, and `yellow`. Although there are a number of additional keywords, there are only 16 original color keywords defined in the HTML 4.01 specification: `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white`, and `yellow`. These 16 keywords are the only ones that are considered valid by the W3C CSS Validation Service.

The world as we see it, however, is not restricted to 16 colors, so why should it be any different in web design? The answer is it's not. We don't need to restrict ourselves to using keywords to specify colors; instead, we can inform the browser we'd like a specific color by using a more specific method of identifying color.

There are a number of methods of selecting colors in CSS; the one we'll be using is **hexadecimal**. "Hexadeciwhat?!" you ask? Good question. Using hexadecimal notation to select colors allows us to access a much broader range of colors, far more than the 16 we've been restricted to so far. But what exactly is hexadecimal?

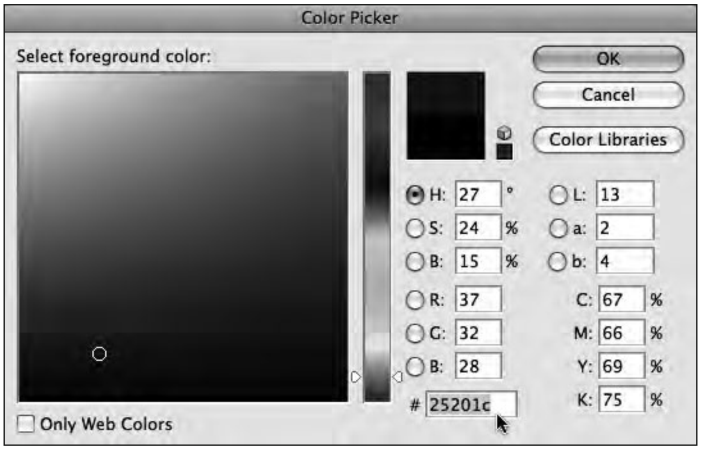
Think of a paint chart. You can select *Blossom Apple White*, a *human-readable* name; however, it will very likely also have a specific reference or code, for example, *DECT31415*, a *machine-readable* name. Let's take a look at the colors we've introduced previously—`teal`, `red`, and `yellow`—and see how they are specified in hexadecimal.

- `teal`: #008080
- `red`: #FF0000
- `yellow`: #FFFF00

Each code indicates a different color, using a computer-friendly method (computers, unlike most humans, love code). Right now you don't need to know the ins and outs of how and why hexadecimal is used, you can simply use the handy page we've created for you at the book's companion web site to select the color you require:

```
www.webstandardistas.com/color
```

Most image editors, for example Photoshop, allow you to find any color's hexadecimal value, as highlighted in Figure 8-7.



H: 27 °
S: 24 %
B: 15 %
R: 37
G: 32
B: 28
25201c
L: 13
a: 2
b: 4
C: 67 %
M: 66 %
Y: 69 %
K: 75 %

Figure 8-7. Photoshop’s Color Picker shows the hexadecimal color value.

You can find further color inspiration at Adobe’s Kuler web site (`http://kuler.adobe.com`), which offers a number of color palettes submitted by users worldwide. Another excellent source of inspiration is COLOURlovers (`www.colourlovers.com`).

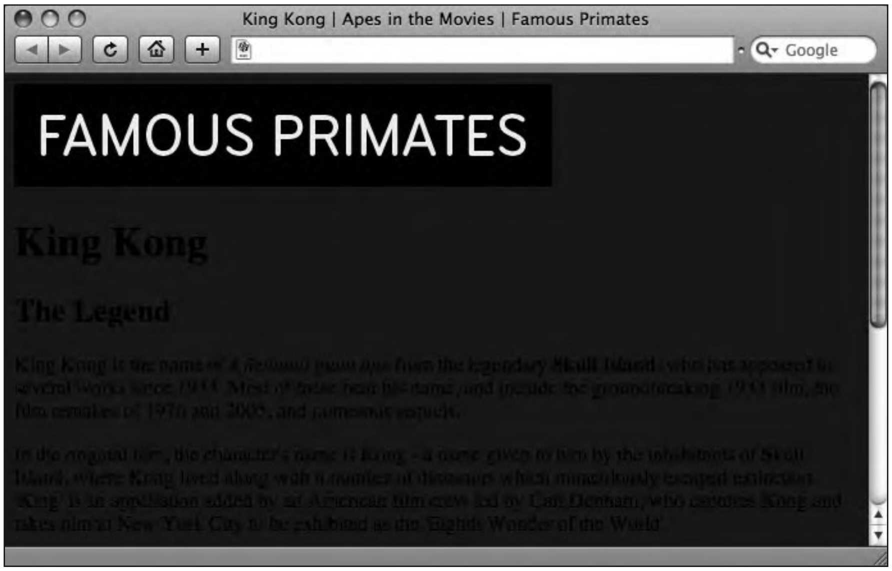
Styling the <body>

The first thing we’ll style is the body; this essentially styles everything that displays within the body element. By now you’re probably a little tired of XHTML pages that display as black text on a white background, courtesy of the browser’s default style sheet. Good news, this is finally about to change.

First we’ll introduce a rule with just one declaration to style the background of the page, in this case displaying it in a dark shade of brown, worthy of a fearsome fictional gorilla. Now that you know you can use hexadecimal color, let’s find a nice shade of dark brown for our King Kong web page. After some deliberation, we’ve chosen #26201C as our perfect shade of brown. Let’s have a look at our page with this first CSS rule added:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>King Kong | Apes in the Movies | Famous Primates</title>
  <style type="text/css">

    body
    {
    background-color: #26201C;
    }
  </style>
</head>
<body>
  <p><img src="primates_brand_black.png" alt="Famous Primates"
  width="420" height="80" /></p>
  <h1>King Kong</h1>
  ...
</body>
</html>
```

The result of adding this simple CSS rule is shown in Figure 8-8.



King Kong

The Legend

King Kong is the name of a fictional giant ape from the legendary Skull Island, who has appeared in several works since 1933. Most of these bear his name and include the groundbreaking 1933 film, the film remakes of 1976 and 2005, and numerous sequels.

In the original film, the character's name is Kong - a name given to him by the inhabitants of Skull Island, where Kong lived along with a number of dinosaurs which miraculously escaped extinction. 'King' is an appellation added by an American film crew led by Carl Denham, who captures Kong and takes him to New York City to be exhibited as the 'Eighth Wonder of the World'.

Figure 8-8. Our King Kong page with a background color added using CSS

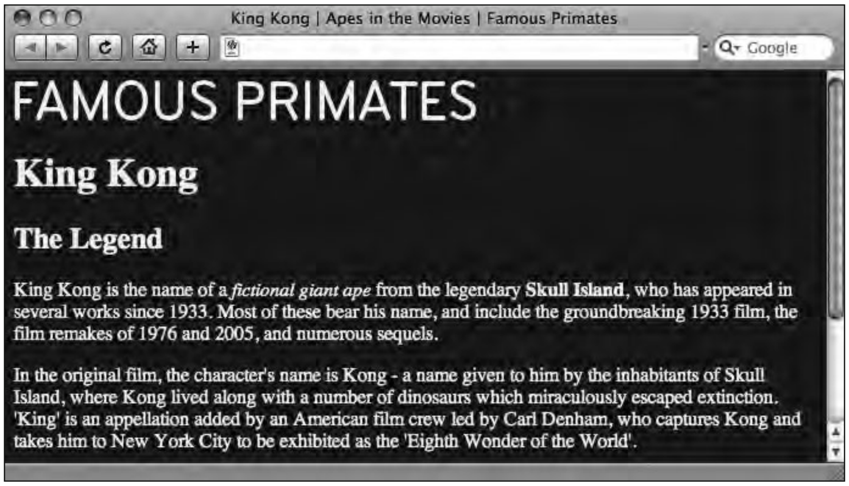
This is a start, but there's clearly more work to do here. As things stand the text on the page is more than a little hard to read when rendered black on dark brown. It's high time for an additional CSS declaration in our body rule:

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

    body
    {
    color: #F3F1EC;
    background-color: #26201C;
    }

  </style>
```

We've added the declaration `color: #F3F1EC;` to our first CSS rule. This sets the text color of all elements appearing within the body to a shade of creamy white. When specifying a `background-color` in your CSS, it's good practice to always specify a contrasting text `color` to ensure that your text remains legible to all.

Another thing we've noticed is that the image displaying the Famous Primates brand would look much better if it matched our fearsome gorilla-brown background color and had some of the space around it removed.

We've taken care of this by firing up our image editor, changing the background color of the Famous Primates brand image to match our CSS background color, `#26201C`, cropping it slightly, and saving a new image to the `images` folder. Our King Kong page now looks like Figure 8-9.



FAMOUS PRIMATES

King Kong

The Legend

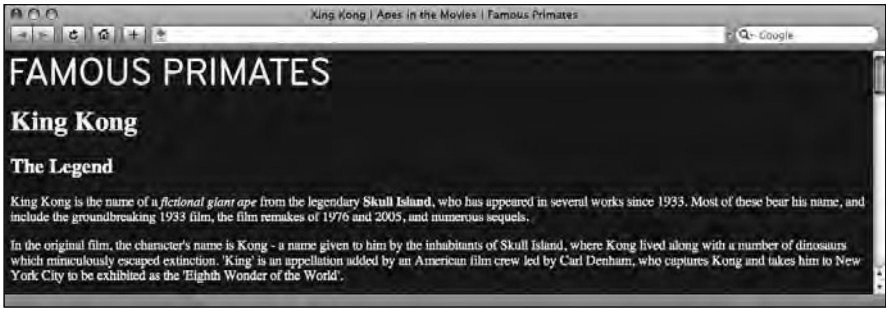
King Kong is the name of a *fictional giant ape* from the legendary **Skull Island**, who has appeared in several works since 1933. Most of these bear his name, and include the groundbreaking 1933 film, the film remakes of 1976 and 2005, and numerous sequels.

In the original film, the character's name is Kong - a name given to him by the inhabitants of Skull Island, where Kong lived along with a number of dinosaurs which miraculously escaped extinction. 'King' is an appellation added by an American film crew led by Carl Denham, who captures Kong and takes him to New York City to be exhibited as the 'Eighth Wonder of the World'.

Figure 8-9. The text color is now a creamy white to contrast with the dark brown background color. The brand image background color has also been changed to match the background color of the page.

Much better!

The next thing we'll tackle is the line length. As things currently stand, our paragraphs occupy the full width of the browser window. For anyone using a large monitor this could result in very long lines of text that are difficult to read, as in Figure 8-10.



FAMOUS PRIMATES

King Kong

The Legend

King Kong is the name of a *fictional giant ape* from the legendary **Skull Island**, who has appeared in several works since 1933. Most of these bear his name, and include the groundbreaking 1933 film, the film remakes of 1976 and 2005, and numerous sequels.

In the original film, the character's name is Kong - a name given to him by the inhabitants of Skull Island, where Kong lived along with a number of dinosaurs which miraculously escaped extinction. 'King' is an appellation added by an American film crew led by Carl Denham, who captures Kong and takes him to New York City to be exhibited as the 'Eighth Wonder of the World'.

Figure 8-10. Without a set width, our paragraphs occupy the full width of the browser window.

Adding a simple declaration to the body rule allows us to control the width of the elements within our page. This will be our first small step toward creating a CSS-based layout that we will expand upon over the next few chapters, introducing more versatile alternatives to setting a width on the body element.

Our CSS rule styling the body now looks like this:

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

  body
  {
  color: #F3F1EC;
  background-color: #26201C;
  width: 550px;
  }

</style>
```

In typography the width of a line of text is known as its ***measure****. Too long a measure results in text that is difficult to read. The eye gets to the end of a very long line of text, and then needs to find the beginning of the next line, often getting lost in the process. Equally too short a measure can be frustrating to read with the eye having to jump back and forth as it reaches the end of lots of very short lines.*

A look at most books will reveal that there is an optimum measure around which most designers tend to settle. This can vary upon the type of content; however, it is generally agreed to be between 50–60 characters per line. In practice different typefaces will result in different line lengths, especially in an online context where the user can scale text up and down.

Our next step is to look at styling our h1 and h2 headings.

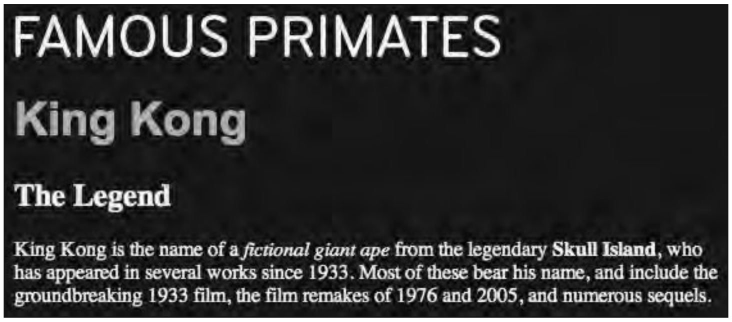
Styling the headings: <h1> and <h2>

Now that we've set the width and styled both the text color and the background color of our King Kong web page, we'll take a look at styling the h1 element. We'll create a CSS rule for it and change its typeface, size, and color.

In the style section of the document, we add a new rule as follows:

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

body
  {
  color: #F3F1EC;
  background-color: #26201C;
  width: 550px;
  }
  h1
  {
  font-family: Arial;
  font-size: 36px;
  color: #9CC4E5;
  }

</style>
```

This additional rule specifies the typeface Arial for our h1. We've also set the size to 36px (pixels), which is slightly larger than the browser default shown in Figures 8-8 to 8-10. The h1 has also been colored a light shade of blue, specified as the hexadecimal color #9CC4E5. The result of these declarations can be seen in Figure 8-11.


FAMOUS PRIMATES

King Kong

The Legend

King Kong is the name of a *fictional giant ape* from the legendary **Skull Island**, who has appeared in several works since 1933. Most of these bear his name, and include the groundbreaking 1933 film, the film remakes of 1976 and 2005, and numerous sequels.

Figure 8-11. The King Kong h1 is now displayed in Arial at 36 pixels in a light shade of blue.

Next we'll add a rule for the h2, almost identical to the h1 rule; it looks like this:

```
h2
{
font-family: Arial;
font-size: 24px;
color: #9CC4E5;
}
```

The only difference between this rule and the previous one styling the h1 is the font-size, which we've set to a slightly smaller size of 24px.

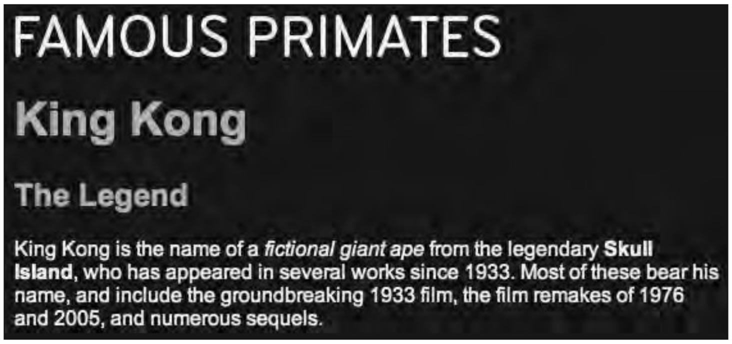
*You'll notice we've used sizes of 36**px** and 24**px**, setting the sizes in px or pixels. There are a number of options for sizing type, including pixels, %, ems, or keywords. We'll introduce the thorny topic of sizing type in more detail in the following chapter when we look closely at styling text.*

As the rules for the h1 and the h2 are largely identical, we can use one of the features of CSS to simplify our style sheet a little. Rather than specifying font-family: Arial; for every single element, we can specify it just once: in the rule styling the body.

The h1, h2, and other elements on the page—which are all nested *within* the body—will *inherit* this style (we'll cover the intricacies of inheritance in the following chapter). Our revised style sheet, with the font-family declaration removed from the h1 and h2 rules and applied to the body rule, looks like this:

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

body
  {
  font-family: Arial;
  color: #F3F1EC;
  background-color: #26201C;
  width: 550px;
  }

  h1
  {
  font-size: 36px;
  color: #9CC4E5;
  }

  h2
  {
  font-size: 24px;
  color: #9CC4E5;
  }

</style>
```

The result of this revision can be seen in Figure 8-12, which shows the headings and the paragraphs now all being displayed in Arial, all thanks to the power of **inheritance**.



FAMOUS PRIMATES

King Kong

The Legend

King Kong is the name of a *fictional giant ape* from the legendary **Skull Island**, who has appeared in several works since 1933. Most of these bear his name, and include the groundbreaking 1933 film, the film remakes of 1976 and 2005, and numerous sequels.

Figure 8-12. Applying the font-family declaration to the body results in our headings and paragraphs all displaying in Arial.

Styling the <p>

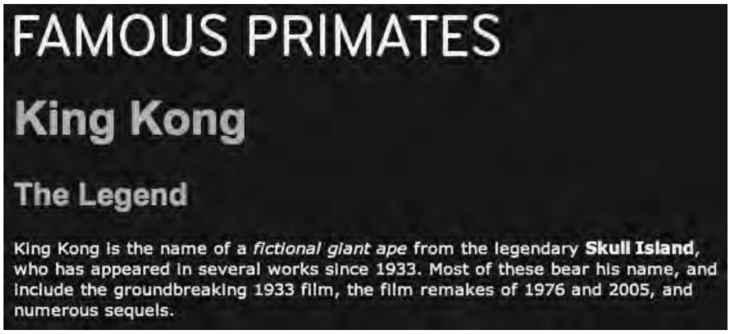
But what if we don't want all headings and paragraphs to be displayed in the same typeface? Have no fear, there's an easy solution at hand. Let's have a look at styling our p, changing its typeface from the style specified previously to a different typeface, Verdana. While we're at it, we'll also change the size of the paragraph text.

We add a rule to our style sheet as follows:

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

  body
  {
  font-family: Arial;
  color: #F3F1EC;
  background-color: #26201C;
  width:550px;
  }

  h1
  {
  font-size: 36px;
  color: #9CC4E5;
  }

  h2
  {
  font-size: 24px;
  color: #9CC4E5;
  }

  p
  {
  font-family: Verdana;
  font-size: 14px;
  }

</style>
```

Adding the font-family: Verdana; declaration to the p overrides the font-family: Arial; declaration of the body. We'll explain exactly how this works in the next chapter when we discuss the topic of **specificity**.

The result of adding this last rule, styling the p, can be seen in Figure 8-13. As you can see, the rule styles all the p elements on the page, displaying them in Verdana at a size of 13 px.



FAMOUS PRIMATES

King Kong

The Legend

King Kong is the name of a *fictional giant ape* from the legendary **Skull Island**, who has appeared in several works since 1933. Most of these bear his name, and include the groundbreaking 1933 film, the film remakes of 1976 and 2005, and numerous sequels.

Figure 8-13. Adding a `font-family` declaration to our `p` overrides the `font-family` declaration on the `body`.

The CSS added to the King Kong page in this chapter is just a start of our journey using style sheets. Although the rules introduced so far are relatively simple ones, they introduce the fundamental principles of CSS. With a little experimentation in your own time, you can try out different typefaces, sizes, and colors and apply CSS to some of the other web pages you've created for your homework.

Commenting your CSS

Just like HTML allows you to use hidden comments within your markup to make notes and selectively hide sections of a web page during the development process, you can also add comments within your CSS, assisting you as you develop your style sheets.

As with HTML comments, CSS comments are useful for a number of purposes: making notes during the development process; commenting out single declarations within a rule; commenting out entire rules; adding structural comments to break your CSS down into logical, grouped sections; and, lastly, giving your CSS a title, useful for keeping a track of the creation date and version number, or noting who created it.

Where HTML uses an opening `<!--` and a closing `-->` to contain your comments, CSS uses an opening `/*` and a closing `*/`. This is shown in the following two examples so that you can clearly distinguish the syntax. The first example shows an HTML comment; you should be familiar with this by now:

```
<!--
<p>I am a paragraph element that has been hidden from the browser
within comments.</p>
-->
```

The next example shows a CSS comment:

```
/*

The following rule has been hidden from the browser within comments.
```

```
h1
{
font-size: 36px;
color: #9CC4E5;
}
*/
```

As we covered in the introduction to this section, comments in CSS can serve a variety of purposes; let's run through these one by one. Although you're only getting started with CSS and some of these uses will be more relevant when you're writing more complicated style sheets and creating external CSS files, we feel it's useful to gather the examples in one section for easy reference.

As with HTML comments, CSS comments can be useful to keep track of changes or to provide notes for useful reference; following is an example of a note:

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

  body
  {
  background-color: #26201C;
  }

  /* All <h1> instances are set to display in Arial at 36px in a shade
  of blue (#9CC4E5) to contrast with the dark brown background. */

  h1
  {
  font-family: Arial;
  font-size: 36px;
  color: #9CC4E5;
  }

  ...

</style>
```

The ability to comment your CSS can also be very useful throughout the development process. As browsers ignore anything between the opening /* and the closing */, CSS comments can be used to selectively switch on and off entire rules, single declarations within rules, or groups of declarations within rules, as shown in the following examples.

Imagine you're creating a style sheet, and in the process you're trying out a new color for your h1s. You're not 100% convinced that the color's right; perhaps the page looked better before you added the color declaration. Commenting out the declaration, as in the following example, allows you to quickly see how the page looked *before* you decided on chimp vomit green for all your h1s:

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

  ...
  h1
  {
  font-family: Arial;
  font-size: 36px;
  /* color: #33FF00; */
  }

  ...

</style>
```

Sitting between comments (or **commented out**), the `color: #33FF00;` declaration will be ignored by the browser, switching off the chimp vomit green on the `h1`s. This method makes experimentation easy; if you want to switch the color back on, all you need to do is uncomment the rule.

Just like cornflower blue, chocolate brown, and school bus yellow are accepted as "officially recognized" color names, so too is chimp vomit green. Coined by noted web standards advocate Jeffrey Zeldman in his book Designing with Web Standards *(Peachpit Press, 2006), the term* chimp vomit green *is used to describe a color generated by a Netscape-related browser bug.*

You can use the same method to temporarily disable more than just one declaration in your style sheet, switching off multiple declarations, or even hiding entire CSS rules as in the next example:

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

  ...

  h1
  {
  font-family: Arial;
  font-size: 36px;
  color: #33FF00;
  }

  /*
  p
  {
  font-family: Verdana;
  font-size: 14px;
  }
  */

  ...

</style>
```

In the preceding example, the comment spans several lines, commenting out the entire p rule. Using CSS comments like this during the development process can be extremely helpful, both when experimenting and troubleshooting.

Although we've only just introduced CSS in this chapter and the style sheets we've shown are quite simple, as your style sheets grow in complexity, you'll find using structural comments to break your CSS down into logical, grouped sections makes the development process easier in the long run. For example, grouping together body and layout sections, typography sections, link styles, and so on can make finding these sections and making changes to them much simpler.

The following examples show some CSS comments used to provide visual breaks within a style sheet, helping to visually separate each logical section. Breaking down a complicated style sheet like this helps finding different sections of the CSS easier:

```
/*
Typography
*/
```

Stylizing these comments a little more can help to visually separate them from your CSS rules as in the following example:

```
/*
Typography
**************************************************
*/
```

Lastly, you can use CSS comments to give your style sheet a title, useful for keeping track of the creation date and version number, or noting who created it. Although we're using an embedded style sheet at this point, CSS comments can be useful to provide you with a quick reference to the status of a style sheet, for example, what version of the style sheet it is and when it was last updated.

The following comment shows a typical example of this that we'll be using at the top of our King Kong web page's CSS as we continue to develop it:

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

  /*
  Famous Primates CSS
  Copyright (C) 2008 Christopher Murphy and Nicklas Persson
  Version: 2.4C
  Last Revision: Thursday, 6 November, 2008 [CM]
  www.webstandardistas.com
  */

  ...

</style>
```

This allows us to keep track of who last edited the CSS and when. In this case we can see the style sheet was last updated on Thursday, 6 November 2008 by CM (the initials of the person who made the last revision).

Summary

So what have we covered? This chapter marked the beginning of our journey into CSS. We've introduced you to some simple CSS rules and covered how to add them to your web pages. Although it might not look like we've changed our basic web page much in the example we've walked through, we've introduced the underlying concept of CSS and how you implement it.

Having followed the preceding examples, you should now know enough to begin adding some style to your well-structured HTML pages that you created as a part of the previous chapters' homework.

In the next chapter we'll take our working knowledge of CSS and expand upon it, enabling you to create web pages that are both well-structured and well-presented. Good times indeed.

Homework: Adding some CSS to Gordo's web page

In this chapter we've introduced CSS, which will allow you to style a number of the elements on the page you've been working on for Gordo. Now it's time for you to give Gordo's page a makeover.

We looked at how CSS rules are written, in particular introducing the idea that one CSS rule could have multiple declarations styling a variety of different properties, for example, `font-family`, `font-size`, `font-weight`, `color`, and `background-color`.

We also showed you how to add an embedded style sheet to a web page where, for now, you'll be locating all of your CSS rules. Lastly, we covered the use of comments in CSS and how they can be used to structure and order your CSS rules.

In our King Kong walkthrough, we styled the `body`, `h1`, `h2`, and `p` elements to introduce you to the fundamental principles of adding CSS to a web page. You'll be applying style to these elements on your Gordo page for this chapter's homework. The process of adding these elements will reinforce the fundamentals of CSS and will form the basis for the following few chapters' homework.

You're embarking on a new language in this homework—writing style sheets. Completing this homework will equip you with a firm grasp of the principles of CSS, something that will prove important as we embark on the forthcoming chapters, which progressively add complexity.

1. Add your <style> tags

Open the gordo.html file in your pioneers folder. In order to add some CSS rules to this page, you'll need to add some <style> tags. In the head section, add an opening <style ="text/css"> tag and a closing </style> tag. You'll be placing all of your CSS rules within these tags.

2. Style the <body>

Referring to the examples in this chapter, style the body element on Gordo's web page. We'd like you to create a rule with three declarations that styles the color and background-color and sets a width for the elements nested within the body.

3. Style the headings

Once you've styled the body, it's time to style the headings. In our King Kong example, we styled the h1 and h2; we'd like you to do the same for the Gordo page. Add new rules to your embedded style sheet for the h1 and h2 elements and style the font-family, font-size, and color of both.

4. Style the paragraphs

The last rule we'd like you to create is for the paragraphs. Create a rule targeting all of the p elements and apply some style to the font-family and font-size to differentiate the paragraphs from the headings.

5. Add a comment

Once you've completed the previous stage, add a CSS comment within the embedded style sheet, giving it a title and a version number (presumably version 1.0), and noting the date you created it. You can do this by leaving a comment at the top of the CSS as follows:

```
/*
Famous Primates CSS
Version: 1.0
Last Revision: Thursday, 6 November, 2008 [CM]
*/
```

To help you with these stages we've created a similarly styled page about King Kong. You can refer to this, using your browser's View Source menu command to see how we've applied our CSS to it, here:

```
www.webstandardistas.com/08/king_kong.html
```

6. Validate

You won't be surprised to hear that we'd like you to validate Gordo's new CSS using the W3C CSS Validation Service. Like the W3C Markup Validation Service—for XHTML—that you've been using so far, it offers an option to validate by direct input, allowing you to copy and paste your code into the web page. Alternatively you can enter the URL of your file if you've uploaded it. The W3C CSS Validation Service is available here:

```
http://jigsaw.w3.org/css-validator/
```

In the event of errors, the CSS validator offers similar clues to its XHTML counterpart, giving the line number of each error. If you run into errors, debug your CSS and revalidate.

Once you're welcomed with the message, "Congratulations! No Error Found." put the kettle on and enjoy a cup of *Kokei Cha* as you prepare yourself for the next chapter.