Since the early days of the Web, one of the biggest challenges when designing Web pages is making them look good in all the different Web browsers in use on the Internet. The creators of Web browsers and the companies that create software for designing Web pages do not always agree on the same standards. That means that even on desktop computers a page that looks great in Firefox 3 may be unreadable in Internet Explorer 6.0 (widely considered one of the most problematic Web browsers on the Internet).

The mobile Web makes designing Web pages even more complicated because so many different types of cell phones and other devices exist and they use different kinds of software.

Most professional designers agree that the best way to create a Web page is to follow the latest in Web standards and use XHTML (eXtensible Hypertext Markup Language, a strict form of HTML) with CSS (Cascading Style Sheets).

In the next few chapters, we focus on the many versions of XHTML and CSS and why you might use one over another for designing mobile sites. But before we describe the variations that work best for mobile use, it’s important to understand the basics of XHTML and CSS.

If you’re already an expert in XHTML and CSS design, I recommend that you at least skim this chapter because the basic concepts covered here are the foundation for the more advanced topics that follow.
Part II: Following Mobile Web Standards

**HTM-What? Exploring HTML and XHTML**

Contrary to popular belief, HTML isn’t a programming language. Rather, it’s a *markup* language: That is, HTML is designed to “mark up” a page, or to provide instructions for how a Web page should look. HTML is written by using **tags**, markup instructions that tell a Web browser how the page should be displayed. For example, to make a section of text italic, you use the HTML tag `<em>`, which stands for *emphasis*. Most tags in HTML include both an open tag and a close tag, indicated by the forward slash `/`. Thus, to make the name of this book appear in italics, I would write the code like this:

```html
<em>Mobile Web Design For Dummies</em>
```

XHTML is a stricter version of HTML and is the recommended language for Web design today. XHTML differs from HTML in several ways. For example, XHTML must be written in lowercase letters but HTML tags can be written in uppercase or lowercase. Also, XHTML, unlike HTML, requires that all tags include a close tag (more on that later in this chapter). All templates and code examples in this book follow the XHTML standard.

To see what the code behind a Web page looks like in most browsers, choose View➪Source. If you’re using Dreamweaver, as shown in Figure 4-1, you can click the Split button (in the upper-left corner of the workspace) to see the code and the design areas of the program at the same time in *split view*.

Split view in Dreamweaver is a useful way to keep an eye on what’s going on behind the scenes, and, as a bonus, it can help you learn a lot of XHTML.

Dreamweaver offers three view options:

- **Code:** In code view, you see only the XHTML and other code.
- **Split:** In split view, the page is divided so you can see the code in one part of the workspace and a view of how the page should be displayed in a Web browser in the other part.
- **Design:** In design view, you see the page as it should appear in a Web browser.

Dreamweaver’s split, code, and design views are integrated, so if you select something in design view, such as the headline you see in Figure 4-1, the same text is highlighted in code view, making it easy to find your place in the code.

If at first glance you think that XHTML code looks like hieroglyphics, don’t give up too quickly. With just a little experience, you can start to recognize common tags, such as the `<h1>` (heading 1) tag that was used to format the headline on the page shown in Figure 4-2.
Chapter 4: Understanding XHTML and CSS

Figure 4-1: Use split view to display both the page design and the code behind the page.

Figure 4-2: A heading 1 tag highlighted in Dreamweaver code view.

To help distinguish the code from the text in a Web page, Dreamweaver displays tags in a contrasting color, usually blue. You can change the size, color, font, and other features of the code in Dreamweaver’s preferences.

Following are a few points to help you better understand XHTML:

- **In XHTML, all tags must include the closing slash.** XHTML tags, even those that stand alone such as the `<br />` tag, must have a close tag, and close tags always contain a forward slash (`/`). For example, the line break tag is `<br>` in HTML, but `<br />` in XHTML.

- **XHTML includes many hierarchical tags.** Examples are the `<h1>` through `<h6>` tags, which are ideally suited to formatting text according to its importance on a Web page. Reserve the `<h1>` tag for the most important text on the page, such as the top headline. `<h2>` is ideal for subheads or secondary headings, `<h3>` for the third level of headings, and so on. A headline formatted with the `<h1>` tag looks like this:

```
<h1>This is a headline</h1>
```
Some tags are complex, and the open and close tags don’t always match. More complicated tags, such as the tags used to create links or insert images into pages, are more challenging to use because they include link information, and the close tag doesn’t always match the open tag. For example, the code to create a link to another Web site looks like this:

```html
<a href="http://www.digitalfamily.com">This is a link to DigitalFamily.com</a>
```

At its heart, XHTML is just text, and believe it or not, you can write XHTML in a plain-text editor as simple as Notepad, SimpleText, or TextEdit. You have to be careful to type all the code perfectly because there is no room for errors or typos in XHTML. After writing code yourself, even to create a simple page, you’re sure to quickly appreciate programs — such as Dreamweaver — that write the code for you.

One of the great advantages of using Dreamweaver is that you can specify formatting by clicking buttons or using menu commands instead of writing the XHTML code. For this and many other reasons, we use Dreamweaver in this book.

## Creating Page Designs with HTML Tables

In the early days of Web design, most page layouts on the Web were created with tables. By merging and splitting table cells, and even adding background images, you could create complex Web designs with tables. CSS expands upon this concept by adding many new design options, including the capability to precisely add margins and padding around elements as well as better control over how and where background images appear.

### How Web browsers work

Web browsers such as Internet Explorer, Firefox, Safari, and most microbrowsers for cellphones are designed to decipher HTML, XHTML, CSS, AJAX, and other code and display the corresponding text, images, and multimedia on a computer screen. Essentially, browsers read the code in a Web page and interpret how the page should be displayed to visitors.

Unfortunately, because Web browsers are created by different companies and the code they display has evolved dramatically over the years, not all Web browsers display Web pages the same way. Differences in browser display can lead to unpredictable (and often frustrating) results because a page that looks good in one browser may be unreadable in another. Add all the mobile devices on the Web and that challenge only gets worse. Much worse. For more information on testing your mobile page design to make sure it looks good to all your visitors, see Chapter 7.
CSS also enables you to keep formatting information separate from content, making it possible to use less code and create pages with smaller file sizes, which download more quickly. Using CSS also makes pages easier to update because you can streamline formatting changes. You can read more about CSS in the next section, “Designing with Cascading Style Sheets.”

Figure 4-3 provides an example of an old-school site created with the HTML `<table>` tag. To help you appreciate how this page was created, I altered the original design to display the table borders. (Most designers turn off table borders to create a cleaner layout.)

If you visit the site at www.chocolategamerules.com, you can see how this same page was created using `<div>` tags and CSS. (I explain how `<div>` tags work within CSS in “Designing with Cascading Style Sheets” later in this chapter.)

Although tables are no longer recommended for creating page layouts, they’re still considered the best way to format tabular data, like the data in a spreadsheet program. In mobile designs, however, it’s important to limit the overall width of tables so that they fit on the smaller mobile screens. For example, you use tables to format a consistent collection of information, such as the photos and scores in the list of winners from the Chocolate Game Rules site shown in Figure 4-4, but when creating a page for the mobile Web, we would recommend you reduce the image size and use narrower columns than those used in the desktop page design.
I understand that many designers still find it easier to create layouts with tables, and not everyone has time to redesign their Web sites right away. However, I recommend using only CSS today for all your Web page layouts, except when you are creating a layout for tabular data. And in those tables, use CSS to add styling, such as background colors or padding.

In mobile Web design it’s even more important not to use tables for layout because many mobile devices don’t render tables well and tables with more than two rows generally won’t fit on the mobile small screens.
Designing with Cascading Style Sheets

The concept of creating styles has been around since long before the Web. Desktop publishing programs (such as Adobe InDesign) and word processing programs (such as Microsoft Word) use styles to manage the formatting and editing of text designed to be printed. In a word processor, you can create and save styles for common features, such as headlines and captions. In Desktop publishing and word processing programs, styles are great timesavers because they enable you to combine a collection of formatting options (such as Arial and bold and italic) into one style and then apply all those options at once to any selected text in your document by using only a single style. The advantage is that if you change a style, you can automatically apply the change everywhere you’ve used that style in a document.

With CSS, you can use style sheets for more than just text formatting. For example, you can use CSS to create styles that align images to the left or right side of a page, add padding around text or images, and change background and link colors. You can also create more than one style sheet for the same page — say, one style sheet that makes your design look good on computers, another for cellphones, and a third for printed pages. For all these reasons (and more), CSS has quickly become the preferred method of designing Web pages among professional Web designers.

Appreciating the advantages of CSS

A Web site designed with CSS separates content from design. Keeping the site content (such as the text and headings) separate from the instructions that tell a browser how the page should look benefits both the designers and your site visitors:

✔ **CSS simplifies design changes.** For example, instead of formatting every headline in your site as 24-point Arial bold, you can create a style for the `<h1>` tag that contains all the formatting information and then apply that style to the text in the XHTML file. You save CSS styles in the header section at the very top of an XHTML page or in a separate file that you can attach to multiple XHTML pages. If you decide later that you want your headlines to use the Garamond font rather than Arial, you simply change the style for the `<h1>` tag once.

✔ **Separating content from design enables you to create different style sheets for different audiences and devices.** In the future, separating content from design is likely to become even more important as a growing number of people view Web pages on everything from giant, flat-screen monitors to tiny, cellphone screens, as shown in Figure 4-5.
Part II: Following Mobile Web Standards

✓ Using CSS makes your site comply with the current standards. Today, the W3C, which sets standards for the Internet, recommends using CSS for nearly every aspect of Web design because the best CSS designs are accessible, flexible, and adaptable.

✓ Pages designed with CSS have cleaner, leaner code. CSS is a more efficient way of designing pages than the HTML attributes we used to use because attributes had to be repeated throughout a page, even when you were adding the same formatting. In contrast, one defined style can be used many times throughout a site. The less code, the faster pages load, which is especially important when you’re designing for mobile devices.

✓ Web sites designed in CSS are accessible to more visitors. Today, there’s a growing movement among some of the best designers in the world to get everyone to follow the same standards, create Web sites with CSS, and make sure they’re accessible to everyone.

When Web designers talk about accessibility, they mean creating a site that can be accessed by anyone who might visit your pages — including people with limited vision who use special browsers (often called screen readers) that read Web pages aloud as well as others who use specialized browsers for a variety of other reasons.
If you work for a university, a nonprofit, a government agency, or a similar organization, you may be required to create accessible designs. Even if you’re not required to design for accessibility, it’s still good to do so because pages that meet accessibility standards also tend to score better in search engines.

**Combining CSS and XHTML**

Most professional Web designers today recommend creating Web page designs by combining XHTML and CSS. How the two work together can be complicated, but you essentially do the following:

1. Use XHTML to create the structure of a page with tags, such as division `<div>`, heading (`<h1>`, `<h2>`, and so on), and paragraph `<p>`.
2. Create styles in CSS that specify the size of these elements, where they appear on a page, and a variety of other formatting options.

Similarly, you use XHTML to insert images and create links, and then add styles to change formatting options, such as removing the underline from your links or changing the color that appears when someone rolls a cursor over a link.

**Creating page layouts with CSS and XHTML**

The key to understanding how CSS works in page layout is to think in terms of designing with a series of infinitely adjustable containers, or *boxes*. This approach to Web design is commonly referred to as the *box model*. First you use HTML tags, such as the `<div>` tag or `<p>` tag, to create a box around your content. Then you use CSS to style each box, controlling the position and alignment of each box with attributes and specifying such settings as margins, padding, and borders.

Although you can use any XHTML tag as part of your page layout, the `<div>` tag is used most often to create the boxes for the main sections of a page, such as the banner area, main content area, sidebars, and footer. The `<div>` tag is like a generic container designed to hold text, images, or other content, or it makes a division on the page that separates one section of content from another. Unlike other XHTML tags, `<div>` has no inherent formatting features. Unless CSS is applied to a `<div>` tag, it can seem invisible on a page. However, the tag has a powerful purpose because any content surrounded by opening and closing `<div>` tags becomes an object (or a box) that can be formatted with CSS. When you create or edit a style that corresponds to a `<div>` tag ID, you can specify properties such as alignment, border, margin, height, and width to control how the `<div>` tag is displayed on the page.
In many CSS layouts, each `<div>` has an ID, which corresponds to a style in the style sheet (although you can use Class styles with `<div>` tags, it’s common to use ID styles for the main `<div>` tags on the page). The ID appears in the XHTML within the `<div>` tag brackets so that the browser knows which style to use to control the formatting of that `<div>` when it displays the page. For example, all the templates have a `<div>` with the ID `container` that controls the overall size of the design area. In the code, the `<div>` looks like this:

```
<div id="container"></div>
```

In the corresponding style sheet, which you can easily access through the CSS panel in Dreamweaver, you’ll find a style called `#container`, which controls the width and other settings for that `<div>`. If this is confusing, don’t worry too much at this stage; my main goal here is to introduce you to the general concepts of XHTML and CSS.

You can find many more lessons on how to create, define, and edit CSS styles in *Dreamweaver CSS For Dummies* (Wiley).

**Understanding style selectors**

When you create a new style, you need to know which selector to use for which job. You can use four main selector types when designing with CSS. If you’re new to working with styles, understanding each selector and its respective restrictions and best uses is a good place to start. The descriptions of each selector in this section can help you understand your options before you move on.

Don’t feel you have to memorize this information. Instead, refer to this list of selectors when you create and edit styles later.

**Class selectors**

The class selector is the most versatile selector option. You can use *class styles* as many times as you want to format any element (from text to images to multimedia on any page in a Web site).

Class style names begin with a period and cannot contain any spaces or special characters (for example, no apostrophes or exclamation points). *Note:* Hyphens and dashes are okay. Thus, you could create a style called `caption` for the text that appears after your pictures, but “my cool captions!” would not be a good name for a class style. Here’s what a class style named `caption` should look like:

```
.caption
```
If you choose class as the selector type and forget to include a period (dot) at the beginning of the name, Dreamweaver adds one for you. Note: Don’t include any space between the dot and the style name.

However, when you apply a class style to text or another element, the dot doesn’t appear in the name when it’s added to your HTML code. Thus, if you applied the .caption style to a paragraph tag to format the text below an image, it would look like this:

```html
<p class="caption">This is a photo of an Egret in flight.</p>
```

Class styles must be applied to an element, such as the paragraph tag shown in this example. You can add class tags to elements that are already defined by other styles.

When you create a class style in Dreamweaver, the style is displayed in the CSS Styles panel on the right side of the workspace (shown in Figure 4-6). You can apply class styles by using the CSS drop-down list, also shown in the figure.

---

**Figure 4-6:** To create a style with a class selector, use the CSS drop-down list.
A common style is one that aligns images and other elements to the right or left of a page. In all our Web sites, we create two such styles to align images to the right and left of any page and name them .float-right and .float-left. These styles typically include margin spacing to create a little white space between an image and text when the text wraps around the aligned image (refer to Figure 4-6).

**ID selectors**

ID styles are the building blocks of most CSS layouts. You should use ID styles only one time per page, making them ideally suited to formatting `<div>` tags and other block-level elements that are used to create distinct sections in a design and appear only once per page.

ID styles must begin with a pound (#) character. Dreamweaver adds # to the beginning of the style name automatically if you forget to include it. As with a class style, don’t include a space between # and the style name.

The ID selector option is a new addition to the CSS Rule dialog box in Dreamweaver CS4. (In CS3, you had to choose the Advanced option to create an ID style.) Like class styles, you can name ID styles anything you like as long as you don’t use spaces or special characters (hyphens and underscores are okay). An ID style used to identify the sidebar section of a page might look like this:

```
#sidebar
```

The # isn’t used in the HTML code when you apply a style to an element, such as a `<div>` tag:

```
<div id="sidebar">Between these tags with the sidebar ID style, you would include any headlines, text, or other elements in your sidebar.</div>
```

In the predesigned CSS layouts included in Dreamweaver, all the designs are created by combining a series of `<div>` tags with ID styles using names such as #container, #header, and #footer to identify the main sections of the design. Figure 4-7 shows a collection of ID and class styles displayed in the CSS Styles panel.

**Tag selectors**

The tag selector is used to redefine existing XHTML tags. Select this option if you want to change the appearance of an existing XHTML tag, such as the `<h1>` (heading 1) tag or `<ul>` (unordered list) tag.
Chapter 4: Understanding XHTML and CSS

Figure 4-7: Styles created with the ID selector are used only once per page and are ideal for creating a CSS layout.

Redefining existing XHTML tags with CSS often has advantages over creating new styles. For example, the Web recognizes content formatted with the heading 1 tag as the most important text on the page. For that reason, many search engines give priority to text formatted with the heading 1 tag. Similarly, the hierarchical structure of the `<h1>`–`<h6>` tags helps ensure that text formatted with the heading 1 tag will be larger relative to text formatted with the heading 2 tag (and so on), even if visitors to your site change the text size in their Web browser.

Because you can change the appearance of headings and other tags with CSS, you can use common XHTML tags with all of the advantages they offer, while still being able to define the font, size, color, spacing, and other formatting options that you prefer in your Web design. When you use the tag selector to create a new style, the style definition you create for each tag is applied automatically to any text or other element formatted with that tag. For example, if you format a headline with an `<h1>` tag and then create a new `<h1>` style, the new style (and its formatting) replaces the original style of the `<h1>` tag.

When you use Adobe Dreamweaver to create a new style, the New style dialog includes a drop-down list where you can choose a tag selector type. If you choose the tag selected, a long list of XHTML tags appears in a drop-down list in the New CSS Rule dialog box. This list makes it easy to select the tag style you want to create, such as the `<h1>` tag shown in Figure 4-8.
Creating compound styles

You use the compound selector to combine two or more style rules. The advantage of compound styles is that you can create style definitions that control the formatting of elements in one part of a page differently from the way they are formatted in another because compound styles only work in places where a defined style is contained within another style. Compound styles are useful when you want to do something like use the heading 1 tag multiple times to format headlines in different ways on the same Web page. For example, you could create one style for headlines that appear in the main story area of a page and another style for headlines that appear in the side-bar on the page but use the heading 1 tag to format both.

To create a compound style, you combine the ID, class, or tag styles. For example:

```
#sidebar1 h1
```

Figure 4-9 shows you how an `<h1>` style defined within a `#sidebar1` ID style looks in the New CSS Rule dialog box. Note that in a compound style, you must include a space between each name or tag in a compound style, and you don’t include the brackets around the tag. In this example, the style definition applies only to `<h1>` tags that appear within another element, such as a `<div>` tag with an ID style `#sidebar1`.

If a compound style combines more than one tag, it’s written like this:

```
#sidebar1 h1 a:link
```
Again, you must include a space between each name or tag. In this example, you see a style that defines the appearance of the active link tag only when the link is located inside an element formatted with the `<h1>` tag inside an element formatted with the #sidebar1 ID. A compound style like this makes it possible to create links that look different in a headline in the sidebar of a page and in another part of the sidebar.

Here’s another example of a compound style that includes styles created with an ID and class selector. This style would specify the way the class style `.caption` should appear only when it is contained within the `<div>` tag styled with the ID #sidebar. Note that there is a space between each style name and that the punctuation used in class and ID styles is maintained.

```plaintext
#sidebar .caption
```

### Understanding rule definition options

In CSS, you have the option of creating internal, external, or inline styles. You can also use a combination of these options, or attach multiple external style sheets to the same Web page. Following is an explanation of these options:

- **Internal styles:** If you create internal styles, the CSS code is stored in the `<head>` area at the top of the HTML page you’re working on, and the styles can be applied only to the page in which they were created. If you’re creating a one-page Web site or creating styles that will be used only on one page, an internal style sheet is fine, but for most sites, external style sheets offer more advantages.
Part II: Following Mobile Web Standards

External styles: If you save your styles in an external style sheet, they’re stored in a separate file with the .css extension. External style sheets can be attached to any or all pages in a Web site in much the same way that you can insert the same image into multiple pages. You can also attach multiple external style sheets to the same page. For example, you can create one style sheet for styles that format text and another for layout styles. You can also create external style sheets for different purposes, such as one for print and one for screen display. One of the biggest advantages of external style sheets is that they make it faster and easier to create new pages, and they make it possible to update styles across many pages at once. Note: You can attach more than one external style sheet to the same Web page.

Inline styles: Inline styles are created in a document where the style is used and applied only to the element it’s attached to in the document. These styles are the least useful of the three style sheet options because any changes to the defined style must be made to the code that contains the element, which means you lose many of the benefits of styles, such as the ability to make global updates and create clean, fast-loading code. For example, creating one style for all your headlines and saving it in an external style sheet is more efficient than applying the style formatting options to each headline separately using inline styles.

At the bottom of the New CSS Rule dialog box, shown in Figure 4-10, you find a Rule Definition drop-down list. Use this list to specify where and how you want to save each new style that you define. The options are

This Document Only: Create an internal style that can only be used in the open document.

New Style Sheet file: Create the new style in an external style sheet and create a new external style sheet simultaneously.

An existing external style sheet: Choose any existing external style sheet attached to the page by selecting the name of the style sheet from the Rule Definition drop-down list (see Figure 4-10, in which the existing style sheet named main.css is being selected).
Chapter 4: Understanding XHTML and CSS

Why so many fonts?

Although you can specify any font you want for text on your Web pages, you don’t have complete control over how that font appears on your visitor’s computer because the font you apply is displayed properly only if your visitors have the same font on their hard drives. To help ensure that your text appears as you intend, Dreamweaver includes collections of the most common fonts on Windows and Macintosh computers, grouped in families, such as Arial, Helvetica, sans-serif, and Georgia, Times New Roman, Times, and serif.

When you apply a collection of fonts like these to your Web page, the browser displays the formatted text in the first font available in the list. For example, if you choose the Georgia font collection and your visitors have Georgia on their hard drives, they’ll see your text in Georgia. If they don’t have Georgia, the text will be displayed in the next font on the list — in this case, Times New Roman — if your visitors have that font. If they don’t have that font either, the text is displayed in Times; if they don’t even have Times (which would be unusual), the browser looks for any serif font. (A serif font, such as Times, has little curly things on the edges of letters; sans serif, such as Arial, means no curly things.)

You can create your own font collections by selecting the Edit Font List option at the bottom of the Font-Family drop-down list in the Property inspector or the Type category of the CSS Rule Definition dialog box. Use the plus and minus buttons at the top of the Edit Font List dialog box, shown here, to add or remove a font collection. To add individual fonts to a collection, select the font name from the bottom right of the dialog box and use the double left arrows to add it to a font list. (Use the double right arrows to remove a font from a collection.)

The only way to ensure that text appears in the font you want is to create the text in a graphic in a program, such as Photoshop or Fireworks, and then insert the graphic with the text into your page. That’s not a bad option for special text, such as banners or logos, but it’s usually not a good option for all your text because graphics take longer to download than text and are harder to update.

Tip

If you’re creating a style that you’ll likely want to use on more than one page in your site, save the style to a new or existing external style sheet. If you save a style in an internal style sheet and later decide you want to add it to an external style sheet, you can move the style by dragging the style into the external style sheet list in the CSS Styles panel.
Looking at the code behind the scenes

Even if you prefer not to look at the code behind your Web pages, it’s helpful to at least have some familiarity with different kinds of tags, CSS, and other code that Dreamweaver creates for you when you design Web pages. The following examples show what the CSS code in an internal or external style sheet looks like in Dreamweaver for the following styles:

- An ID style created with the ID selector named #container and defined as 960 pixels wide with the left and right margins set to auto (a cool trick for centering a CSS design).

```css
#container {
  width: 960px;
  margin-right: auto;
  margin-left: auto;
}
```

- A style created with a class selector named .caption and defined as Verdana, Arial, Helvetica, sans serif, small, italic, and bold.

```css
.caption {
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: small;
  font-style: italic;
  font-weight: bold;
}
```

- A style created with a tag selector to redefine the HTML tag <h1> as Arial, Helvetica, sans serif, large, and bold. (Note: Because the heading tags already include bold formatting, it’s not necessary to include bold in the style definition.)

```css
H1 {
  font-family: Arial, Helvetica, sans-serif;
  font-size: large;
}
```