



# Essential CSS: What You Need to Know

**E**xpression Web contains one of the best CSS editors available. Before you can create styles in Expression Web, you must first learn about how stylesheets work. After an overview of CSS, it will be time to create styles and apply them to your web pages. In this chapter, you will learn about selectors, attributes, and values. In the following chapters, you will learn how to use CSS to define the presentation of your page using the tools in Expression Web.

## What Is CSS?

*Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g. fonts, colors, spacing) to web documents.*

<http://web4.w3.org/Style/CSS/>

Stylesheets contain rules that allow you to control how you would like your document displayed in your visitor's web browser. You can manipulate just about anything from fonts to colors to graphics to positioning using a set of simple rules. For example, you might decide that you want to have all your headings in a document displayed in Arial font. To create this effect, you apply a rule that looks like this:

```
h1 { font-family: Arial, Helvetica, sans-serif; }
```

The previous rule consists of the following parts:

- *Selector:* This is the HTML element, class, or ID (more on classes and IDs later). In this case, it is a `<h1>` header element.
- *An opening curly bracket:* This starts the style definition for the selector.
- *A declaration:* This is the part that is inside the curly brackets instructing the browser how to display the selector when it is used in the HTML. The declaration is divided into two sections:



With stylesheets, you can change the look of every heading 2 and heading 3 on the page by editing the style definition in one place if you are using an external stylesheet linked to the web pages in your site.

In Chapter 4, you learned how to use HTML in Expression Web to provide the semantic structure for the exchange of information; simply put, this means using headings, subheadings, paragraphs, lists, and data tables to aid the visitor in reading the page.

CSS takes that structure and presents it in an attractive visual manner, which is referred to by web professionals as separating content from presentation. HTML is for structuring the page content, and CSS is for page display.

The advantages of stylesheets are as follows:

- You'll gain control over the appearance and format of your web pages.
- By using the same stylesheet on each page of your site, the pages will remain consistent.
- It'll be easy to maintain a web page or website.
- In most cases, your file size will be considerably smaller than a table-based layout and as a result load faster for your site visitors.
- If your browser does not understand the stylesheet, the page will still display using the browser defaults. This means that your page will degrade gracefully in a variety of devices and older browsers.

## Before Stylesheets

In the early days, designers used HTML to provide structure to a document without being able to control its presentation in the browser; for example, you could do the following:

- Create heading and subheadings
- Define paragraphs
- Emphasize text as either italic or bold
- Create links in other sections of the document or other pages
- Insert images
- Add tables

As HTML evolved, additional elements were added, so you could then do the following:

- Center text
- Align text
- Change text colors
- Choose the font face
- Change the text size

Then designers started using tables to format pages. Tables were created to display data, but once web designers, most of whom in the mid-90s started their careers in print, discovered that using tables for layouts gave them control over the page's look closer to that of a printed page, tables took over web design.

This abuse of table markup led to large files, bloated code, and the delusion that you can have pixel-perfect pages across a variety of browsers and platforms.

Consider a page that has `<table>` and `<font>` tags and was created with a WYSIWYG HTML editor like the website generators offered by many hosting companies, such as Geocities. Some of these applications do not even combine the `<font>` tag and might not even close HTML elements properly. For example, it was not uncommon to end up with something like this at the beginning of every paragraph:

```
<p>
<font face=Arial>
<font size=+2>
<font color="blue">
  Your text here
</font>
</font>
</font>
</p>
<p>
<font face=Arial>
<font size=+2>
<font color="blue">
  More Text here
</font>
</font>
</font>
</p>
```

If you were to type the font code, you would write it like this:

```
<p>
<font face="Arial" size="+2" color="blue">
  Your text here.
</font>
</p>
<p>
<font face="Arial" size="+2" color="blue">
  More text here.
</font>
</p>
```

If you use stylesheets instead, you would write the same presentation information in your external stylesheet as follows:

```
p {
  font-size: x-large;
  font-family: Arial, Helvetica, sans-serif;
  color: #0000FF;
}

<p>Your text here.</p>
<p>More text here.</p>
```

The paragraph style is defined only one time, and you do not have to remember to format every paragraph. If later you decide that you want your `<p>` elements to be `medium` instead of `x-large`, all you need to change is *one line* in the stylesheet instead of changing the HTML of every paragraph in the page or website.

## Which Browsers Support Stylesheets?

The following browsers support stylesheets:

- All version 5 or later browsers
- All versions of Safari, Firefox, and Mozilla
- Version 4 of Opera and Internet Explorer
- Netscape Navigator 4 to a limited degree

## Types of Styles

Three types of `<style>` blocks are available with CSS. In Chapter 3, when showing how to configure the CSS tab of the Page Editor Options, I touched upon the first two. The three options are as follows:

- *Inline*: This affects only the single instance of an element.

```
<p style="color: red">Red Text </p>
```

- *Document level*: This is the default when “(classes)” is selected in the Page Editor Options and is sometimes referred to as an embedded stylesheet.

```
<style type="text/css">
p {color: red;}
</style>
```

- *External*: This is where your style definitions are contained in a separate CSS page that is linked to one or more pages on your website.

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

## Inline

This type of style is closest to the older `<font>` tag method of styling since it applies only to the section of the page enclosed by the style declaration. To create a gray heading 2 that is centered on the page using an inline style, you use code that looks like this:

```
<h2 style="color: gray; text-align: center;">
  Centered Gray Heading
</h2>
```

The `<h2>` will display like Figure 5-1.

## Centered Gray Heading

**Figure 5-1.** *Inline style example*

This works well in every browser that has even a modicum of stylesheet support, but it has the same maintenance drawbacks of using `<font>` tags.

## Document Level

If you want all `<h2>` elements to be centered and gray on one page only, the style code can go in a `<style>` block in the `<head>` section of the page instead of writing the style for each line:

```
<style type="text/css">
h2 {
  color: gray;
  text-align: center;
}
</style>
```

When you use Expression Web to create your styles with the configuration options recommended in Chapter 3, you will get a code block that looks like the one shown in Figure 5-1. The `type="text/css"` part tells the browser that the code inside the `<style>` element is CSS. To have a web page that validates, you must include the type of style, which Expression Web will add automatically for you.

When you define a style in the `<head>` section of your document, the style will apply to the selected elements on the page. By defining `h2` in the `<head>` section, every instance of `<h2>` will be red and centered without additional code.

## External

The final type of stylesheet is an external page that has the `.css` file extension. This is nothing but a plain-text page with just the CSS code on the page. The code is simply the style definition without the enclosing style block as follows:

```
h2 { color: red; text-align: center; }
```

You attach the stylesheet to the web page using either the `link` or the `@import` method.

To link a stylesheet to a web page, you use the following code:

```
<link href="basic.css" rel="stylesheet" type="text/css" />
```

This is the standard method of attaching a stylesheet to a web page, but there is an alternative method where you embed the stylesheet in the <head> section of the page inside a <style> block. To embed the stylesheet, you use this code:

```
<style type="text/css"> @import url("../site.css");</style>
```

There are two reasons you might want to use the insert method over the link method:

- To hide styles from older browsers that might have a problem with certain CSS 2 elements. The primary browser in this case is Netscape 4.x, which has rather buggy HTML support. Embedding is not often used for this purpose because Netscape Navigator 4.x is rarely used anymore.
- To change the order of the cascade so that the external stylesheet is treated as if it were a document-level stylesheet.

## C for Cascade

So far I have addressed only the style portion of CSS, but the first letter in CSS stands for cascading, or how the styles flow down from one level of the document like the cascade of a waterfall. Understanding the cascade, a.k.a. how style conflicts are resolved, is an important part of using styles. As stated, there are three types of styles: inline, document level, and external. The cascade is important because you can have more than one style that applies to an individual element. The style definition closest to the code that uses the style is the style that will be applied in terms of both specificity and the location of the style definition. This means you can define a default for your website in the external stylesheet, change that default in the <head> section for just one page, or change it in the element itself to force a difference in just one instance. The best way to understand how the cascade works is to use styles; for that reason, it is important that you do Exercise 5-1.

---

**Note** Each of the figures in Exercise 5-1 use the same Internet Explorer 7 window size.

---

### Exercise 5-1. Understanding the Cascade

In this exercise, you'll get some practice using styles so you can understand how the cascade part of styles works.

1. Copy the following code to a new page, and save it as `external.css` in your local website:

```
body {  
    background-color: #FFFFFF;  
    color: #000000;  
    font-family: Tahoma, Helvetica, Arial, sans-serif;  
}
```

```

h2 {
  color: gray;
  text-align: center;
}
p {
  font-size: small;
}

```

2. In the same folder as you saved the external.css file, copy the following code to a new HTML page, and save it as cascade.html:

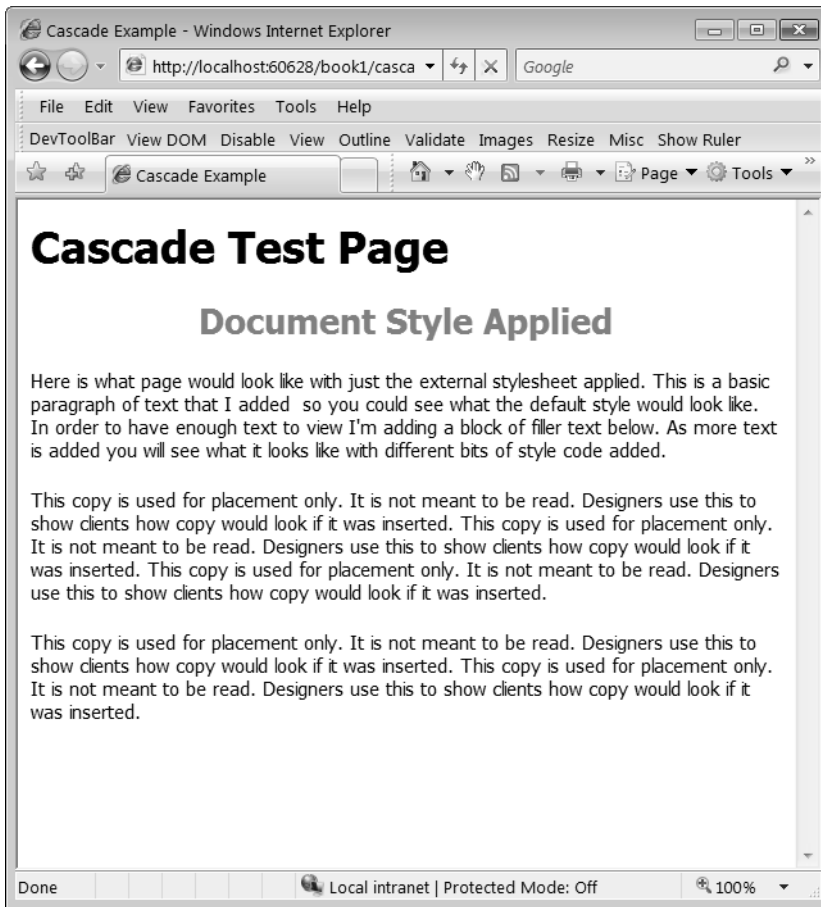
```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Cascade Example </title>
<link href="external.css" rel="stylesheet" type="text/css" />
</head>
<body>
<h1>Cascade Test Page</h1>
<h2>Document Style Applied</h2>
<p>
  Here is what the page would look like with just the external stylesheet
  applied. This is a basic paragraph of text that I added so you could
  see what the default style would look like. In order to have enough
  text to view I'm adding a block of filler text below. As more text
  is added you will see what it looks like with different bits of
  style code added.
</p>
<p>
  This copy is used for placement only. It is not meant to be read.
  Designers use this to show clients how copy would look if it was
  inserted. This copy is used for placement only. It is not meant
  to be read. Designers use this to show clients how copy would
  look if it was inserted. This copy is used for placement only.
  It is not meant to be read. Designers use this to show clients how
  copy would look if it was inserted.
<p>
  This copy is used for placement only. It is not meant to be read.
  Designers use this to show clients how copy would look if it was
  inserted. This copy is used for placement only. It is not meant to
  be read. Designers use this to show clients how copy would look if
  it was inserted.
</p>
</body>
</html>

```



3. When viewed in a browser, the page should look like Figure 5-2.



**Figure 5-2.** Browser display when an external stylesheet is linked to this web page

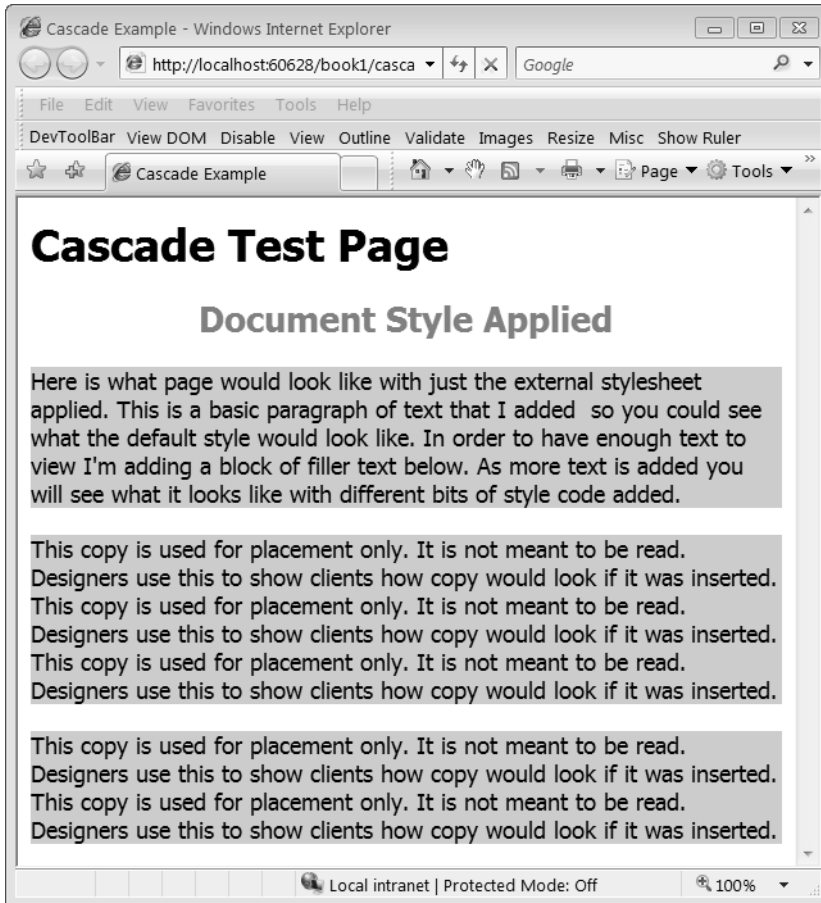
The line of code that attaches the external stylesheet is as follows:

```
<link href="external.css" rel="stylesheet" type="text/css" />
```

4. Next, add the following `<style>` block to the `<head>` section of your web page just below the `<link>` element:

```
<style type="text/css">
p {
  background-color: #cccccc;
  font-size: medium;
}
</style>
```

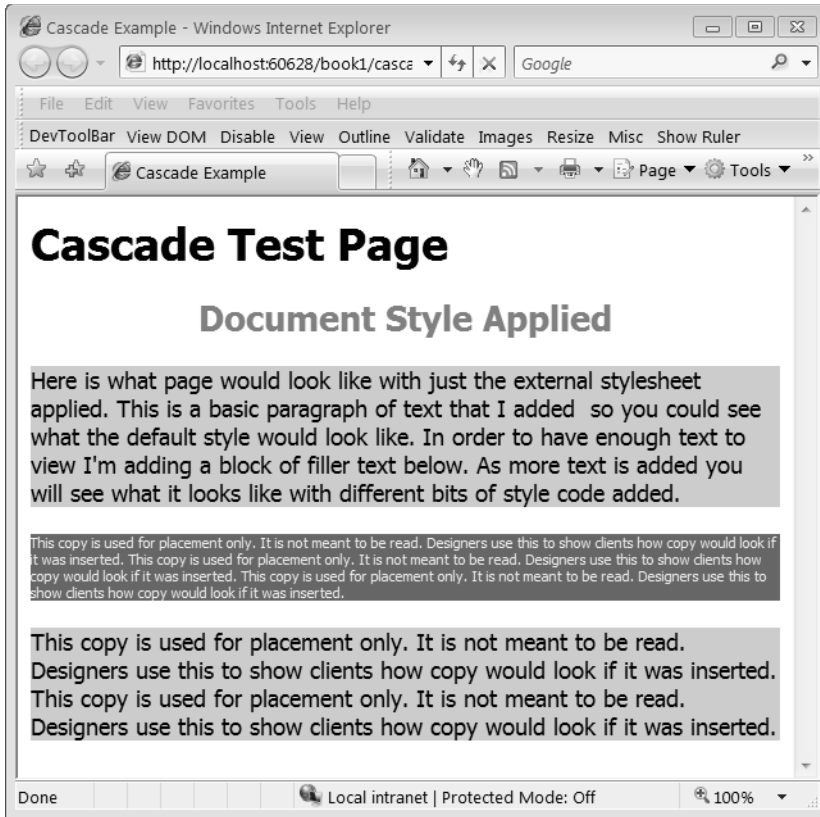
- The page will now look like Figure 5-3; notice how the background color of the paragraph text is now gray and larger than in the original example? The paragraph text size changed from small in the external stylesheet to medium in the <head> section's <style> block. In case you can't tell the text size difference just by looking at the image, the browser size remained unchanged from Figure 5-2, yet the number of words on each line is much less.



**Figure 5-3.** The <style> block has replaced the external stylesheet size and has added a background color.

- Next, add an inline style to the middle paragraph. Change the <p> tag to this:  

```
<p style="background-color: #666666; font-size: x-small; color: #eeeeee;" >
```
- Then preview it again in the browser; it should look like Figure 5-4:



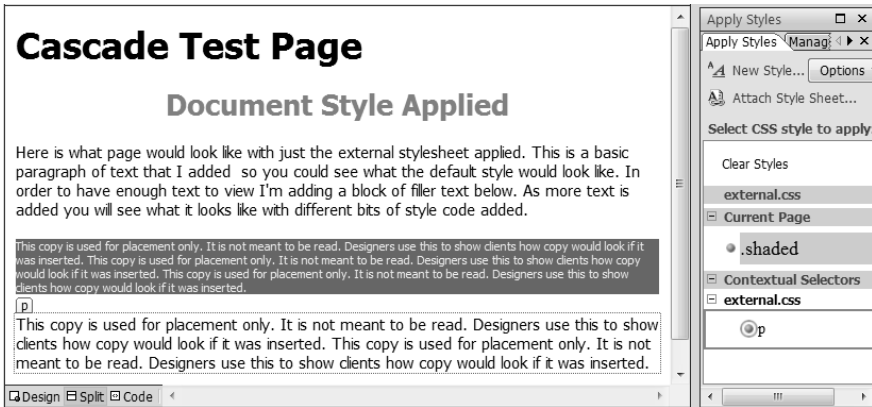
**Figure 5-4.** Notice that the inline style overwrites both the `<head>` section and the external style in the middle paragraph.

Notice that the center paragraph with the inline style applied has `x-small` silver text on a dark gray background.

8. One last change: in the `<head>` section, change the `<p>` element style to a class by replacing the `p` with `.shaded` (that is a dot in front of `shaded`) so that the style reflects the function of the class and is not tied to a specific HTML element:

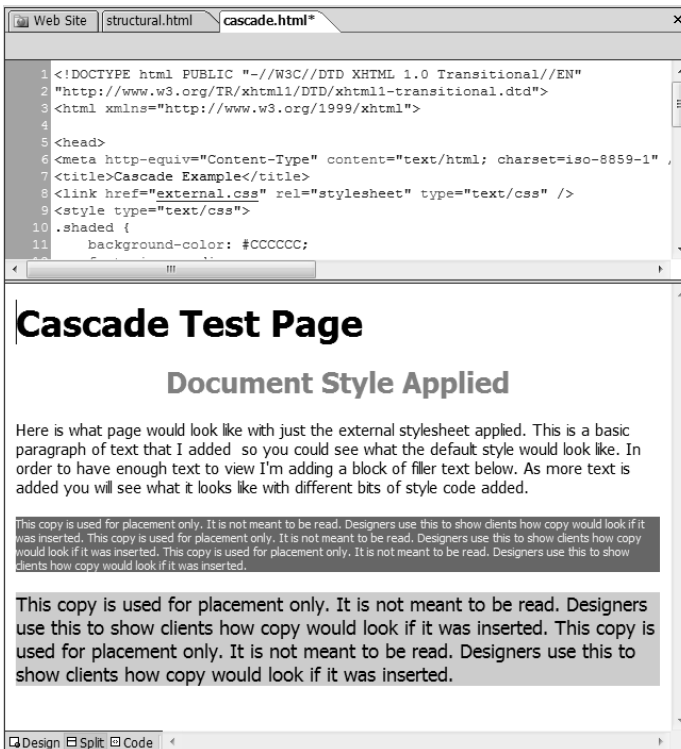
```
<style type="text/css">
.shaded {
    background-color: #cccccc;
    font-size: medium;
}
</style>
```

9. Now in Design view, your paragraphs with the gray background should be white again. Click anywhere in the last paragraphs, and from the Styles drop-down list, use the bottom option, Apply Styles, which will open the Apply Styles task pane. Double-click the entry shown for the shaded class, as shown in Figure 5-5.



**Figure 5-5.** Double-clicking the style in the Apply Styles task pane will apply the style.

10. Your page should now look like Figure 5-6.



**Figure 5-6.** You can now see each of the styles applied.

In Figure 5-6, you see the external style for paragraphs applied to the first paragraph. The second paragraph has an inline style. *You should use inline styles sparingly.* The third paragraph has a class from the <head> section applied. Don't forget to preview the page in your browser.

## @import and Older Browsers

When an external stylesheet is embedded using `@import` in the page instead of being linked, the effect is the same as if the style itself was written in the `<style>` block where the stylesheet is imported. This gives an external stylesheet the same weight as writing the code directly in the document.

This method is not supported by some older browsers and allows you to send one set of styles to modern web browsers and a simpler set to older browsers; one is linked to the page and contains simple styles that even version 4 browsers support, and one is imported and contains enhanced features that allow finer control or effects that modern browsers support. Remember that the syntax used to import a stylesheet is as follows:

```
<style type="text/css">
<!-- @import url("eternal.css"); -->
</style>
```

---

**Note** The `@import` statement is sometimes surrounded by HTML comment tags, such as `<!-- inside this is a comment that will not be displayed in the web browser -->`, to prevent older browsers that do not understand `@import` from showing in the code. In most cases, this is no longer necessary.

---

## Specificity and Inheritance

You have to consider another factor when determining which style definition will apply to your content: inheritance. Elements that are contained inside another element will inherit the styles applied to the parent. For example, every web page must start with `<html>`, and for the content to display in the browser, it must be inside the `<body>` element. This means every item that will display on the page inherits the style definition for the `<html>` and `<body>` elements.

In the earlier examples, the paragraph elements and the `<h1>` all inherited the font family specified in the `<body>` element definition. If you were to change the `<body>` element's definition of `font-size: medium` to `font-size: x-large`, you would see that the `<h1>` would have a corresponding increase in size.

When you look at the `<h2>` element's position in the document structure hierarchy, you will see that the `<h2>` is as follows:

```
<html> <body> <h2>
```

---

**Tip** You can quickly check an element's position in the page hierarchy using the Quick Tag bar at the top of the document window.

---

This means the `<h2>` element is the closest HTML element with a style definition to the page content in the browser; as a result, the styles defined for the `<h2>` will be applied whenever

there is a conflict with the style definitions for the `<body>` or `<html>` element. In the earlier examples, there is a conflict between the black specified in the `<body>` element's style and the gray specified in the `<h2>` element's style:

```
body {  
    background-color: #FFFFFF;  
    color: #000000;  
    font-family: Tahoma, Helvetica, Arial, sans-serif;  
}  
h2 {  
    color: gray; /* the h2 color specified conflicts with the body*/  
    text-align: center;  
}
```

Under the cascade, the `<h2>` will display as gray since the `<h2>` element is wrapped around the content.

## Cascade Summary

As you can see in the previous examples, the cascade means the style closest to the HTML code takes precedence in the case of a conflict. In summary, the browser renders styles in the following order, moving to the next level if it does not find a definition until it reaches the browser defaults:

1. Inline style
2. Document-level style, including embedded external stylesheets
3. External style using link relative
4. Browser defaults

To determine specificity, look at the document structure.

## Class and ID

So far, in the examples, I have primarily used HTML elements as selectors. Using just HTML elements would be very limiting since there can be only one definition per element that applies to all elements on a page without the use of inline style code. Inline styles would not be an improvement over using font tags and would not allow you to have different styles based on context.

Many web designers and usability experts find that establishing blocks or areas on their web pages for specific types of content makes the pages more attractive and easier to use for most users. A style for the site's menu is a common example of using a distinct style for one section of the page.

As I briefly touched on in Chapter 3 when configuring the CSS tab of the Page Editor Options, you have two additional attributes without semantic meaning that you can use to define and apply your styles. These are `class` and `id`.

## Class

What is a CSS class?

- A class always begins with a period in your style definition like the `.shaded` class you created earlier.
- You can use a class multiple times on a page.
- You apply class to an HTML element using a `class` attribute, such as `class="class_name"`. For example, this is a class applied to the `<p>` element: `<p class="notice">`.
- A class's name must begin with a letter but might contain numbers. Do not use spaces in class names.

## ID

What is a CSS ID?

- An ID always begins with a number sign (`#`) in your style definition such as `#navbar`.
- You can use an ID *only once* on any individual page in your site.
- You apply an ID to an HTML element using the `id` attribute, such as `id="id_name"`, without the number sign. For example, this is an ID applied to the `<ul>` element: `<ul id="menu">`.
- An ID's name must begin with a letter but might contain numbers. Do not use spaces in the ID name.

## `<div>` and `<span>`

Although these two items are HTML elements, they are very important to the proper use of CSS. These two elements are almost as important as either a class or an ID. When you use a `<div>`, you selectively apply CSS to specific sections of your page using contextual selectors so that you do not have to apply a class or ID to each individual element but instead to the container only. Once you begin using CSS for page layout in the next chapter, you will be using `<div>` elements to create the different sections of your web page and apply different style definitions to the links depending on whether they are in the main content area, navigation, footer, and so on, without applying classes to each link.

A `<span>`, on the other hand, applies to a section of text inside an HTML element. This allows you to apply a style inline to change a section of a paragraph or other content without changing the items that immediately surround it.

## Setting Properties and Values

When you write a style definition, you can write it all on one line, break it into multiple lines, or use any combination of the two. In most cases, it does not matter what order you write properties and values in, but in a few cases the order makes a difference. In those cases, I will note the correct order when I explain the property.

### Page Elements

In most cases, the first element you will want to define is the overall style of the page. You can do this in one of two ways, either by defining the `<html>` element or by defining the `<body>` element. You will want to place your primary definitions into the `<body>` element since there are some issues in older browsers with less-than-stellar CSS support when it comes to styling the `<html>` element.

In this section, you will look at the most commonly used properties:

- `margin`
- `padding`
- `background-color`
- `color`
- `font`

### BROWSER DISPLAY DEFAULTS

Before you start adding style properties and values, you should know what the default display of an element is because you do not need to specify the defaults you do not want to change. You need to specify only the properties you want to change from the defaults:

- All headings default to bold, and the browser determines the size; generally, h1–h3 are larger than body text, while h4–h6 are smaller.
- Normal is the default for body text, all list items, and any other nonheader elements.
- The browser determines the default font, which is usually a serif font except for the `<code>` element, which defaults to a monospace font.
- There will be space between the edge of your content and the side of the browser; some browsers will use `margin` (Firefox, Internet Explorer, and Safari) and others `padding` (Opera).
- Left is the default alignment for all elements.

I'll discuss browser defaults in more detail when relevant to the CSS you will be using in each section.



## Margin and Padding

Every browser has defaults for each of these properties defined in the browser's DOM, and like most browser defaults, they are not the same in every browser. For example, every browser will have some default amount of spacing between the content and the edges of the browser screen. In Internet Explorer, Firefox, and Safari, the space is controlled by the `margin` attribute. In Opera, it is controlled by the `padding` attribute. Exercise 5-2 shows how to tweak some of the browser defaults.

### Exercise 5-2. Changing the Browser Defaults

In this exercise, you'll change some of the browser defaults.

1. Copy and paste this code, and view the resulting page in Firefox/Internet Explorer or Safari and Opera. Make sure the code is formatted exactly as shown when you look at the HTML, complete with the line breaks and spacing used inside the `<div>` with the `codeblock` style applied.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1" />
  <title>Default Browser Test</title>
  <style type="text/css">
  body {
    background-color: #B5FF84;
    margin: 0;
  }
  p {
    background-color: #ffffff;
  }
  .codeblock {
    margin-left: auto;
    margin-right: auto;
    padding: 10px;
    border: 1px solid #000000;
    background-color: #FFFCC;
    width: 22em;
    white-space: pre;
  }
</style>
</head>
```

```

<body>
  <h1>Browser Test</h1>
  <p>This page has the following style code:</p>
  <div class="codeblock">
    <code>&lt;style type="text/css"&quot;&gt;
      body {
        background-color: #B5FF84;
        margin: 0;
      }
      p {
        background-color: #ffffff;
      }
      .codeblock {
        margin-left: auto;
        margin-right: auto;
        padding: 10px;
        border: 1px solid #000000;
        background-color: #FFFCC;
        width: 22em;
        white-space: pre;
      }
    &lt;/style&gt;
  </code>
</div>
<p>In Firefox, Internet Explorer, and Safari the text on this page
should be obnoxiously close to the side of the browser, which makes
it hard to read. In Opera this page should have normal default
browser spacing between the edge of the browser window and the text,
which makes it easier to read. </p>
<p>You might also notice a font display difference as well. If you
have access to more than one operating system, please view this
page on the different systems to see what difference the default
font settings make. </p>
<p>The codeblock div above demonstrates the effect of padding
and margin. To make the difference easier to see, a border was used
to outline the div area. The margin is the space on the left
between the edge of the page and the beginning of the div,
while padding is the space between the content and the border.</p>
</body>
</html>

```

2. Save this page as `browsers.html`, and preview it in Firefox, Internet Explorer, and Opera, as shown in Figure 5-7.
3. To make a page appear the same in all browsers, both the `margin` and the `padding` need either to be set to 0 or to be set explicitly in the code.



**Figure 5-7.** The browsers from left to right are Firefox 1.5, Internet Explorer 7, and Opera 9.

4. Make sure you read the page created since there is more information on margins and padding in that page.
5. Change the `<body>` style code to read as follows, and view the page again to see the effect of adding padding: `0` to the style definition:
 

```
body { background-color: #B5FF84; margin: 0; padding: 0; }
```
6. Preview in your browser, and compare the differences in display.

Some CSS properties will accept multiple values in one line. Table 5-1 shows these properties.

**Table 5-1.** There Are a Limited Number of CSS Properties That Accept Shorthand

Property	Values Accepted
font	family, size, weight, style, variant, line height
background	color, image, repeat, attachment, position
border, border-top, border-right, border-left, border-bottom	size, style, color
list-style	type, position, image
outlines	color, style, width

Some properties accept multiple values that apply to different sides of the element:

- border-style
- border-width
- border-color
- margin
- padding

Table 5-2 shows how the values are applied.

**Table 5-2.** *The Order of Application*

Number of Values	Order of Application
One	Applied to all four sides
Two	Top/bottom, left/right
Four	Top, right, bottom, left

## Color

How you use color on your web page is one of the first elements most people will notice when they visit your site.

### Background Color

You should have noticed when you previewed the `browsers.html` page that I set a background color for the page of light green and different background colors on other elements so that you could see how `margin` and `padding` work. I did this using the following:

```
background-color: #B4FF84;
```

Depending on what browser you are using, when no background color is specified, you saw either white (most modern browsers) or gray, which is used by a minority of browsers, mostly older browsers such as Netscape Navigator 4. Since browsers do not always default to the same background color, you cannot rely on the browser to always render a white background. Regardless of whether the browser defaults to white, some people set their background colors to something garish to test whether the background color has been specified, but why do they do that?

One reason is as a reminder to always set their own page's background color. Others set a color preference because they prefer not to have a white background and choose to apply a user stylesheet.

To see what color your browser renders the background, comment out the `background-color` property and value by changing the body background color to look like this:

```
/*background-color: #B5FF84;*/
```

---

**Note** To comment out a portion of your stylesheet, you place a backslash and asterisk (`/*`) where you want the comment to begin and the opposite (`*/`) where you want it to end.

---

## Text Color

The text you saw in your browser was most likely black, but the text could have been a different color because the text color was not specified in your stylesheet. According to the W3C Web Content Accessibility Guidelines, you should always specify a text color when you specify a background color. To do that, you use `color: #000000;` in your style.

The colors you choose need to have sufficient contrast to make it easy for visitors on a wide variety of monitors to be able to read your page. Remember that there is a significant difference in gamma (color intensity) between Mac and PC monitors. There is also the issue of color temperature difference between cathode ray tubes (CRT) vs. liquid crystal display (LCD), not to mention the problems associated with older monitors. In addition, many web visitors have monitors that are poorly calibrated, and their color rendering can be very off. Never assume because the contrast is sufficient on your machine that it is equally clear on other people's computers.

If it is possible, try to view your page on at least a CRT *and* an LCD monitor, even if you can't view it on both the Mac and Windows platforms. When you do not have the option of viewing your pages in a variety of monitors, look at the color values, and make sure you are not using two mid-range colors such as red on gray or two shades of gray, or your visitors might not want to stay on your site. Try the following code in your web page; I have seen pages using this style on the Web:

```
<p style="background-color: #eeeeee; color: #999999; padding: 9px;
font-size: 8px; ">
What do you think of this text?
</p>
```

Would you want to read a web page where the content looked like that? If you do not have access to other computers, you can use color tools such as those offered by VisiBone (<http://www.visibone.com>) and Juicy Studio (<http://juicystudio.com/services/colourcontrast.php>).

Another option is to join web design mailing lists, newsgroups, or forums and ask, "How does my page render on a (whatever you need a check on)?"

## Color Syntax

Color is used for elements other than just backgrounds and text. You can also specify border colors. No matter where you are specifying the color, you need to write the color in a way that the browser can understand. The specifications give three methods to tell the browser what color to use:

- Named color
- Hex color
- RGB color

## Named Color

Browsers that support CSS 2+ allow up to 216 named colors to be used. You can find these named colors along with their hex equivalents at [http://www.w3schools.com/css/css\\_colornames.asp](http://www.w3schools.com/css/css_colornames.asp).

---

**Note** Some browsers do not support all the colors, and some are not consistent in applying the same color value to the same name.

---

CSS 1 recognizes only the 16 web-safe color names of Aqua, Black, Blue, Fuchsia, Gray, Green, Lime, Maroon, Navy, Olive, Purple, Red, Silver, Teal, White, and Yellow.

You can also use `transparent` for the absence of color. Note that if Netscape 4.x legacy support is important, you should avoid using `transparent` in a style definition, or you will get a strange color. (In days gone by, people would type their name for the color and view it in Netscape to see what color they were. I seem to recall my name produced a rather unattractive shade. If you want to find out what color you are, you can download and install Netscape 4.x from the browser archive at <http://browsers.evolt.org/>.)

The syntax for using named colors is as follows:

```
color: black;
background-color: white;
border-color: red;
```

## Hex Color

Hex colors are the web native way to specify colors. A hex color consists of three pairs of 0–9 or a–f values to match the values of red, green, and blue in the color. For more information on hex colors and how to convert between hex and RGB, see [http://en.wikipedia.org/wiki/Hex\\_triplet](http://en.wikipedia.org/wiki/Hex_triplet).

The syntax for using hex colors is as follows:

```
color: #000000;
background-color: #ffffff;
border-color: #ff0000;
```

## RGB Color

RGB (which stands for red, green, blue) is an additive color model where the primary colors of light are blended to create all other colors. Since computer screens display color by emitting light, this is a better approach than CMYK, which is a subtractive method of creating color. In addition, many people who come to web design from a graphics background are more comfortable working in RGB instead of hex.

The syntax for using RGB colors is as follows:

```
color: rgb(0,0,0);
background-color: rgb(255,255,255);
border-color: rgb(255,0,0);
```

Although you might use any of the three methods for specifying color, it is best to decide on one naming convention and use it.

---

**Tip** It is best to specify the hex value. RGB is for print, and named colors are unpredictable.

---

## Background Properties

Several properties affect how the background of your page is displayed, as described in the following sections.

### The background-image Property

You can specify an image to use behind your `<html>` element using the `background-image` property, but you can use a background image on any element, including `<div>` and `<span>`. By default, `background-image: url("image.jpg");` will tile your image across the page or other element, repeating both horizontally and vertically.

### The background-repeat Property

Sometimes you will not want your background image to repeat or to repeat in only one direction. You can change the default as follows:

- `background-repeat: repeat-x;` will repeat the image horizontally but not vertically.
- `background-repeat: repeat-y;` will repeat the image vertically but not horizontally.
- `background-repeat: no-repeat;` will prevent the image from tiling.

### The background-attachment Property

`background-attachment` allows you to specify whether you want your background image to scroll with your page. You have two choices:

- `scroll` is the default, which means the image will move with your page as you scroll.
- `fixed` is used when you want the image to stay put and have the text scroll across the top of the image in the background. The perfect example for this would be a company logo or watermark.

Unfortunately, Internet Explorer 6 and earlier will honor only `background-attachment: fixed` on the `<body>` element. Other browsers might render `fixed` differently as well. You can experiment with it on your own if you want. The syntax is `background-attachment: fixed;`

## The background-position Property

At one time, having a stationary background image that served as a watermark was trendy, but it is now out of fashion. If you want to see how a watermark looks, use `background-position: center;`

`center` tells the browser to place the image 50 percent horizontally and 50 percent vertically on the page.

To position your background somewhere other than the default top left or centered on the page, you use a combination of keywords. Table 5-3 lists the keyword pairs and their positions.

**Table 5-3.** *CSS Keyword Pairs and Their Values*

Keyword Pair	Horizontal Position	Vertical Position
top left	0%	0%
top center	50%	0%
top right	100%	0%
center left	0%	50%
center center	50%	50%
center right	100%	50%
bottom left	0%	100%
bottom center	50%	100%
bottom right	100%	100%

Alternatively, you can specify the position using pixels, as in `background-position: 100px 100px;`

## Background Shorthand

You can group all the elements together using `background: list of values`. When using shorthand for background properties, the order is not important. You can declare the values using one to five properties. I do not recommend using shorthand for the background properties unless you are using at least two values. For example:

```
body { background:#efefef url(css.jpg) no-repeat fixed left; }
```

That would be the same as this:

```
body {
  background-color: efefef;
  background-image: url(css.jpg);
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-position: left;
}
```



## UNITS OF MEASUREMENT

No matter what you are doing, you will need to specify a size, whether it is a width, height, border, or some other element/attribute value. You might use one of two methods:

- *Length property value:* The length value is a number that you use to specify where you want something to be placed in a browser. It can be a measure in pixels, inches, and so on, and it can have a plus or minus sign in front of the number to indicate how much you want something removed from the previously specified position. Each of the units is represented by a two-letter abbreviation, as shown in Table 5-4.
- *Percentage value:* The percentage value allows you to specify a portion. You must include a number with or without a +/- sign and end it with a % sign.

**Table 5-4.** *CSS Units of Measurement*

Name	Abbreviation	Type
centimeter	cm	Absolute.
em	em	Relative to the visitor's font size setting on their computer using the printer's em x/y ratio of the capital <i>M</i> of the selected font.
ex	ex	Similar to the em except using the x/y ratio of the lowercase <i>x</i> .
inch	in	Absolute.
millimeter	mm	Absolute.
percentage	%	Relative either to the page default or to a parent setting.
pica	pc	Absolute printer's setting of 12 points.
pixel	px	While technically pixel is device-dependent, screen-dependent, and not related to print, on Internet Explorer it is treated as an absolute unit of measurement.
point	pt	Absolute printer's setting of 1/72 of an inch but in reality device and operating system dependent.

Notice that the absolute units of measurement are print units of measurement with the exception of the pixel unit. Unless you are creating a print stylesheet, you should stay away from picas, inches, and metric units of measurement. Even in a print stylesheet, you should use these units of measurement with caution since you do not necessarily know the size of paper your visitor will be using. They could be using U.S. letter or European A4 or some other paper size.

Although pixels work well for margins, padding, and borders, neither pixels nor points always work well for text sizes since they do not scale in Internet Explorer; stick with em, keyword, or %.

■ **Caution** Every time you create a style with a property that has a size value, you must specify the unit of measurement immediately after the number. If you use 10 without specifying px, pt, %, or em, the browser must guess whether you mean percent, pixel, point, or em. Unlike in HTML, there is no default size unit in CSS.

Exception: The only exception to the rule that you must specify a unit of measurement is when specifying the value 0, because zero is always zero no matter what you are measuring.

---

## Hyperlinks and Pseudo-Classes

Hyperlinks have a unique set of default values. Links by default are blue and underlined to distinguish a link from ordinary text. The HTML specification and usability and accessibility guidelines state that links must always be identifiable as hyperlinks and different from ordinary text. Hyperlinks have an indication state that shows whether you have visited the linked page or whether the link has focus. You can apply separate styles to each of those states using pseudo-classes.

---

■ **Note** The underline you see on hyperlinks is not the equivalent of the old HTML element `<u>`. To remove the underline, you would use `text-decoration: none;`. Do not remove the default underline unless you have provided clear differences between hyperlinks and ordinary text.

---

### Pseudo-Classes

There are four pseudo-classes for links:

- `link`: The default state before the link has been used.
  - `visited`: The state that indicates the linked page is in the browser cache.
  - `hover`: The change of style when the visitor mouses over the link. The pseudo-class `focus` performs the same function for keyboard users.
  - `active`: The style that displays between the time you click the link and when the linked page is loaded. Most people will never see this style if your pages load quickly.
- 

■ **Note** This is one case where order matters. If you put the `hover` definition before the `visited` definition, you will not get the `hover` style when the visitor mouses over the link.

---

### Pseudo-class Inheritance

The pseudo-classes do not inherit from each other, but they do inherit from the anchor element `<a>` when no pseudo-class is specified or when the `<a>` definition overrides the pseudo-class default. For example, if you write the following style declaration:

```
a {color: red; text-decoration: none;}
```

you will find that the link will continue to be red and without an underline even after the linked page has been visited. If you add the following selector definition:

```
a:hover {text-decoration: underline;}
```

the text will remain red with a red underline on mouseover. Exercise 5-3 shows how to change the default so that visitors to your page see an underline when they mouse over a link.

### Exercise 5-3. Seeing Pseudo-classes in Action

In this exercise, you'll add an underline when a visitor mouses over a link.

1. Type the following code on a new page, save it, and preview it in your browser:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1" />
  <title>Link Test</title>
  <style type="text/css">
    a {
      color: #666666;
      text-decoration: none;
    }
    a:visited {
      color: blue;
    }
    a:hover {
      text-decoration: underline;
    }
  </style>
</head>
<body>
  <p><a href="http://google.com">Google</a></p>
</body>
</html>
```

2. Next, use the link to visit Google, and then use your Back button to return to your page.
3. Refresh to force the page to reload.

Most browsers default to purple for visited links; with the code used in this exercise, your link will revert to blue, but the `hover` will still add an underline. If you had placed the `visited` pseudo-class after the `hover`, the underline would not appear when you moused over a visited link.

## Writing a Custom Class for a Hyperlink with a Pseudo-Class

Usually when you write a custom class, all you need to do is start the name of the class with a period. Pseudo-classes require a different syntax. If you want a link to have a custom class, you need to use the syntax `a.class:pseudo`. The most common reason for custom classes on links is when you are linking to a page outside your website and want to alert the visitor that the link is external to your site or will open in a new window. You'll do just that in Exercise 5-4.

### Exercise 5-4. Using Custom Classes and Pseudo-Classes

In this exercise, you'll see how custom classes and pseudo-classes work.

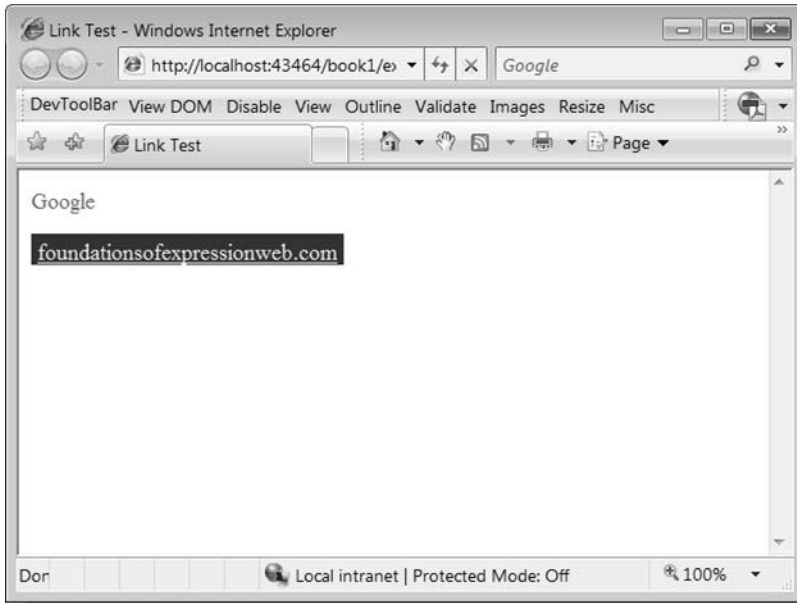
1. To create custom class links in your stylesheet, add the following code to the page you created in the previous exercise:

```
a.offsite {
  color: #eeeeee;
  background-color: #333333;
  text-decoration: underline;
  padding: .3em;
}
a.offsite:visited {
  text-decoration: none;
  background-color: #COCOCO;
}
a.offsite:hover {
  text-decoration: underline;
}
```

2. Then in the `<body>` section, add this:

```
<p><a href="http://foundationsofexpressionweb.com" class="offsite"
  title="following this link will take you off our site">
  foundationsofexpressionweb.com</a></p>
```

3. Now, preview the page in your browser again to see the differences between the two links, as shown in Figure 5-8.



**Figure 5-8.** *A default link style and a class applied to a link*

Make sure you mouse over each link to see the behaviors as well as the differences after you have visited the link site.

---

## Applying a Declaration to a Group of Selectors

You might decide that you want to have all elements of a certain type display with the same base definition. A typical example would be to use a different font for your `<h1>`, `<h2>`, and `<h3>` headings from the other text on the page. To achieve this, define multiple selectors with the same property values group, and separate the selectors with commas. For example:

```
h1, h2, h3 {  
    font-family : Arial, Helvetica, sans-serif;  
    color : #000080;  
}
```

You can still specify other properties, such as `font-size`, individually without needing to repeat the font and color definition in each style:

```
h1 { font-size: 1.5em; }
```

Exercise 5-5 shows how to style multiple selectors.

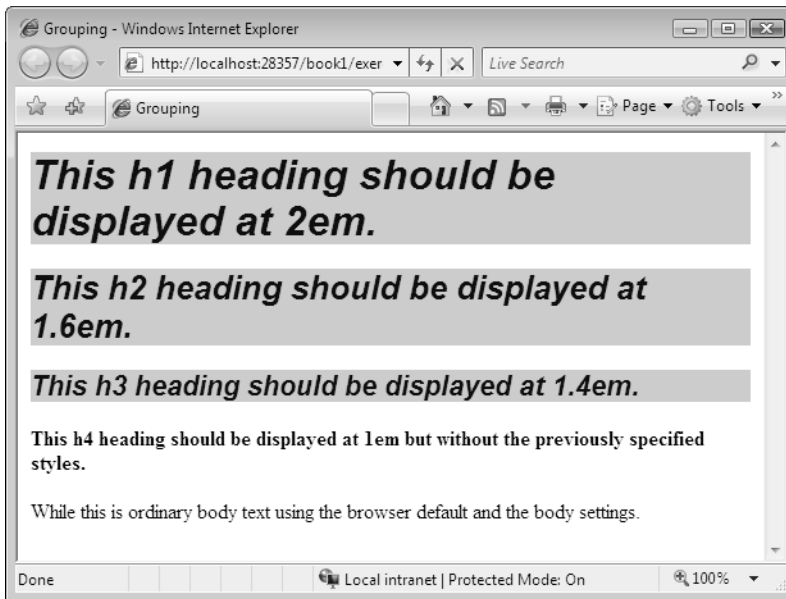
## Exercise 5-5. Styling Multiple Selectors

In this exercise, you will specify the font family, style, and color along with a background color for your `<h1>`–`<h3>` elements. Then you will specify the size of each of the `hx` (`x` stands for the heading level) tags individually.

1. Create a new web page with the following code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Grouping</title>
  <style type="text/css">
  body {
    color: black;
    background-color: white;
  }
  h1, h2, h3 {
    font-family : Arial, Helvetica, sans-serif;
    font-style : italic;
    color : #000080;
    background-color : #cccccc;
  }
  h1 {font-size: 2em;}
  h2 {font-size: 1.6em;}
  h3 {font-size: 1.4em;}
  h4 {font-size: 1em;}
</style>
</head>
<body>
  <h1>This h1 heading should be displayed at 2em.</h1>
  <h2>This h2 heading should be displayed at 1.6em.</h2>
  <h3>This h3 heading should be displayed at 1.4em.</h3>
  <h4>This h4 heading should be displayed at 1em but
  without the previously specified styles.</h4>
  <p>While this is ordinary body text using the browser
  default and the body settings. </p>
</body>
</html>
```

2. Save your page as `grouping.html`, and preview in your web browser. You should see the page shown in Figure 5-9.



**Figure 5-9.** `<h1>`, `<h2>`, and `<h3>` all display the common background and color but the individually set text size.

When you are grouping your selectors, make sure you use a comma to separate them, or otherwise you will be creating what is called a contextual selector.

## What Is a Contextual Selector?

A contextual selector is a style that is applied based on its location in the code. The selector uses CSS inheritance and specificity to provide a different style to part of your page based on the HTML hierarchy and is best demonstrated with an example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1" />
  <title>Contextual Selectors</title>
  <style type="text/css">
  #masthead {
    background-color: #5B5B5B;
    color: #FFFFFF;
    margin: 0;
    padding: .2em 0;
  }
</style>
</head>
<body>
  <h1>This h1 heading should be displayed at 2em.</h1>
  <h2>This h2 heading should be displayed at 1.6em.</h2>
  <h3>This h3 heading should be displayed at 1.4em.</h3>
  <h4>This h4 heading should be displayed at 1em but without the previously
specified styles.</h4>
  <p>While this is ordinary body text using the browser default and the
body settings.</p>
</body>
</html>
```

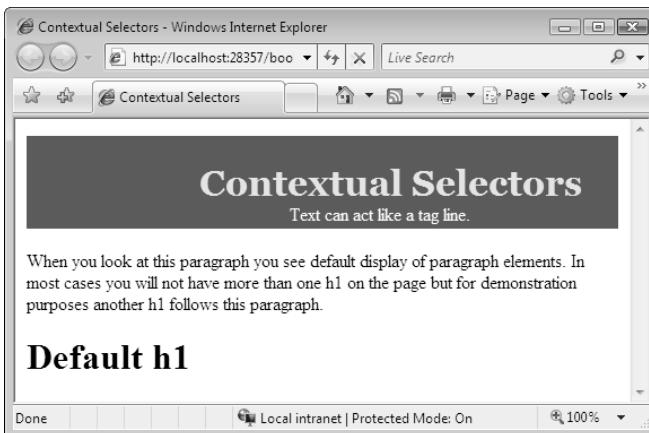
```

#masthead h1 {
  color: #E4E4E4;
  font-family: Georgia, "Times New Roman", Times, serif;
  margin-bottom: 0;
  padding-right: 1em;
  text-align: right;
}

#masthead p {
  margin-top: 0;
  text-align: right;
  width: 75%;
}
</style>
</head>
<body>
  <div id="masthead">
    <h1>Contextual Selectors</h1>
    <p>Text can act like a tag line.</p>
  </div>
  <p>When you look at this paragraph, you see default
  display of paragraph elements. In most cases, you will
  not have more than one h1 on the page, but for
  demonstration purposes another h1 follows this paragraph. </p>
  <h1>Default h1</h1>
</body>
</html>

```

When you preview this page in your browser, you will see the difference between the `<h1>` that is inside a `<div>` with the ID of `masthead` and the `<h1>` that is not, as shown in Figure 5-10.



**Figure 5-10.** The two `<h1>` elements look different depending on their locations in the code without using separate classes.



## Fonts

Most of the properties available for fonts are clear from the property names, and Expression Web has excellent tools for helping you with your font family groups, as you saw on the Tools ► Page Editor Options ► Font Families tab in Chapter 3. Instead of an exercise or example of fonts here, you will look at the options as you can use them to create styles in Expression Web later in this chapter.

## Box Properties

I have touched on the issue of how margins, padding, and borders are treated differently between browsers. How height, width, padding, border, and margins behave is referred to as the box model. How a browser calculates the size of a box can make a significant and visible difference in how the page is rendered. This box model issue is one of the reasons why I emphasize that the Web is not print. When you print a document, it will look the same to every visitor because you can control the paper, the ink, and other variables. On the Web, your visitor controls the operating system, browser, screen resolution, and screen calibration and can break any web page.

When you hear or see the term box properties, it will generally refer to a subset of the properties of block-level elements. The relevant properties are as follows:

- margin
- padding
- border

## Changing the Document Flow with CSS

When your page displays in the order that your HTML is written and you have not used position or float attributes, your page is said to use a flow layout. So far all the HTML pages used in this book's exercises have been flow layouts rendering in the order of the source code.

You can move content to change the presentation to something more visually appealing in more than one way; what each method has in common is that one or more elements are moved from the natural flow of the document. Two methods, float and position relative, are influenced but not displayed in the location in the HTML source code where the property/value is used. The location in the code is the reference point for where the file will display. The other method, position absolute, is independent of the source code order with the top-left corner being the default point of reference. There are advantages and disadvantages to all layout or position methods, which means you need to select the one that best suits your needs.

## Floats

Although technically not a positioning element, float still removes the content from its normal document flow position. float is similar to the HTML align property. The content that is floated moves to the left or right of the surrounding content, which will then wrap around the floated content. This means the other content will be aware of and react to the placement of your float.

Instead of using the `vspace` and `hspace` attributes along with the HTML `align` attribute, you use padding or margins in your style definition to add space between the flow content and your floated content. `float` has three possible values:

- left
- right
- none

---

**Note** `none` is the default, but you should not use `float: none`; unless you are changing the property back from floating to not floating.

---

Floats are removed from the natural document flow but are not treated the same as the `position` property. Sometimes you'll want to ensure that some element is positioned after the floated item and not beside it. You restore the natural document flow by using the `clear` property. This forces the content to start after a floated object instead of around it. Its values are as follows:

- none (the default)
- left
- right
- both

## Position Absolute

`position: absolute` has two uses:

- Overlapping items on the page for effect
- Page layout

`position: absolute` was one of the earliest positioning methods supported by browsers, but there are limits to what you can do with it reliably across browsers if you want to create more than very simple layouts.

The properties available when you use `position: absolute` are simple enough to understand:

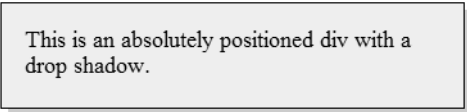
```
position: absolute;
top: 10px;
left: 10px;
bottom: 10px;
right: 10px;
```

These properties behave exactly as you would expect by their names with the exception of `bottom`. Whether the bottom is calculated from the bottom of the web page or the viewport (browser window) depends on the browser. This makes it difficult to create a standard layout with a footer using absolute positioning. Other issues with using `position: absolute` for page layout are related to the box model and/or the uncertainty of exactly how much height an individual block element will have. Exercise 5-6 shows how to overlap elements on your web page.

### Exercise 5-6. Overlapping for Effect

The other use for `position: absolute` is to create overlapping effects. Some people will use this technique to create drop shadows and other effects on their pages.

By using absolute positioning, you are able to specify precisely where the element will be displayed on your web page. In Figure 5-11 all of the elements are positioned using pixels.



This is an absolutely positioned div with a drop shadow.

**Figure 5-11.** *Overlapping for effect can be effective, but you must check how the effect renders in a variety of web browsers.*

The style used to produce the drop shadow effect shown in the figure is as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1" />
  <title>Overlapping for Effect</title>
  <style type="text/css">
    .box {
      width: 360px;
      height: 81px;
      position: absolute;
      top: 50px;
      left: 50px;
      z-index: 100;
      border: 1px black solid;
      background-color: #eeeeee;
    }
  </style>
</head>
</html>
```

```
.box p {
  padding: 1em;
}
.shadow {
  width: 360px;
  height: 81px;
  top: 56px;
  left: 56px;
  position: absolute;
  background-color: #cccccc;
  z-index: 1;
}
</style>
</head>
<body>
  <div class="box">
    <p>This is an absolutely positioned div with a drop shadow.</p>
  </div>
  <div class="shadow">
    &nbsp; </div>
</body>
</html>
```

Try the following changes, and preview each one to see the effect:

1. Change the width of the shadow class.
  2. Change the height of the shadow class.
  3. Remove the z-index from the box class.
- 

---

**Note** Some browsers will collapse a `<div>` or other element that has no content. You can prevent that from occurring by using a nonbreaking space written as `&nbsp;` in your code. Expression Web will insert one for you when you use the spacebar in Design view.

---

## Position Relative

`position: relative` can be a difficult concept to grasp, because unlike `float` it doesn't move in one direction relative to its HTML source code; however, it's not totally removed from the document flow like `position: absolute` either.

`position: relative` moves an element by the amount specified by the `top`, `left`, `right`, and/or `bottom` properties. All the other elements still flow as if the relatively positioned element was not moved. These properties are similar to the absolutely positioned settings, but they work in a slightly different manner.

The value for these four positions is from the position the element would have been in if no positioning was applied, while absolute positioning measures from the top left of the page or other container it is inside:

- `top`: Defines how far from the top of its normal position the top of the container will appear. If a positive value is used, then the container is moved down from the normal position, while a negative value will move the container up from the normal position.
- `left`: Defines how far from its normal position the left of the container is to appear. Positive values will move the container right, and negative values will move it left.
- `right`: Defines how far from its normal position the left edge of the container will appear, not the right edge as you would expect. Setting `right: -5px` is the same as setting `left: 5px` with relative positioning.
- `bottom`: Behaves in a similar manner to `right` in that it defines how far from the normal position the top of the container will appear. Setting `bottom: 50px` is the same as setting `top: -50px`.

In Figure 5-12, an image was placed in the top-right corner to show where the content with the gray background would normally begin inside the first container.

---

**Note** The image has a class to place it at the starting point for demonstration purposes only.

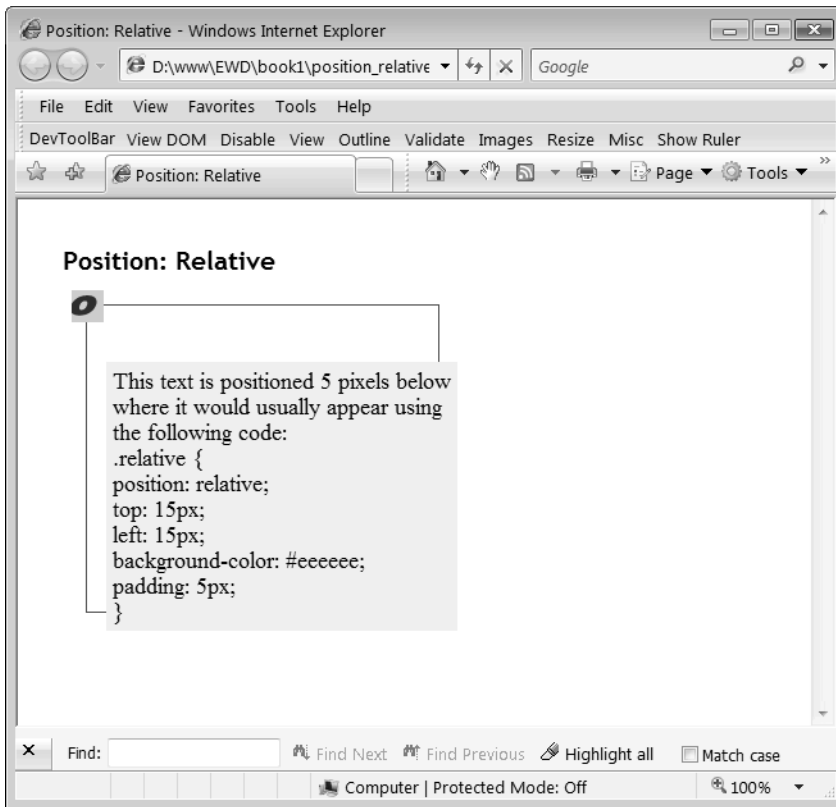
---

The second `<div>` is located inside a container with the following style:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Position: Relative</title>
  <style type="text/css">
  body {
    margin: 2em;
  }
  h3 {
    font-family: trebuchet ms,Arial,sans-serif;
  }
  p {
```

```
    margin: 8px 5px;
}
.container {
  border: green 1px solid;
  width: 50%;
  top: 0px;
  left: 0px;
  margin: 1em;
  z-index: 10;
}
.relative {
  position: relative;
  top: 15px;
  left: 15px;
  background-color: #eeeeee;
  padding: 5px;
}
.normal {
  z-index: 20;
  Position: relative;
  top: -12px;
  left: -12px;
}
</style>
</head>
<body>
  <h3>Position: Relative</h3>
  <div class="relative">
    This text is positioned 5 pixels below where it would usually appear
    using the following code: <br />
    .relative {<br />
      position: relative;<br />
      top: 15px;<br />
      left: 15px;<br />
      background-color: #eeeeee;<br />
      padding: 5px;<br />
    }
  </div>

</body>
</html>
```



**Figure 5-12.** This example uses `position: absolute` to mark the top left of the container with an image.

Notice that the relatively positioned code goes outside the container that it is inside by the amount of the offset.

## Summary

The CSS I have covered in this chapter has introduced you to the selectors, properties, and values available in CSS. It is not a comprehensive guide but the first steps in understanding the effects of the CSS you will be creating in the next two chapters as you begin using the tools that Expression Web offers to help you to create an attractive website that will also meet your company's needs.

Pixel-perfect display is virtually impossible on the Web. For extremely precise placement, you need print. Instead of fretting over that loss of control, focus on the features you can create on the Web that you cannot do in print such as hyperlinks.

In the following chapter, you will be using Expression Web's CSS tools, which will help you create attractive and functional websites.