

Cascading Style Sheets

Introduction

Cascading style sheets (CSS) is a standard defined by the World Wide Web Consortium that offers designers more flexibility and accuracy when defining the appearance of text and formats than standard HTML. Essentially, CSS allows designers to manipulate the appearance of the webpage without affecting its HTML structure (Offer, 2000). For example, if you wanted to change all the text in your document to blue, and all the headlines to green, with standard HTML one would have to manually change the elements on the page one by one. By using CSS, it's possible to redefine the body elements in the entire document to turn blue with just one instruction, and then perform another step to change the headlines to green (Green, 2002).

Significance of Cascading Style Sheets

If used appropriately, cascading style sheets can be quite beneficial to the web designer. In addition to text, cascading style sheets can define spacing between lines, the size of the type in pixels instead of points, and define specific fonts within pages. Formatting each individual page becomes very cumbersome for the designer, especially if their site contains hundreds of pages. With style sheets, one only needs to specify such preferences *once*, and the style can be applied to an entire site. And if the designer decides to change the width of the page, then he or she only needs to change this preference in *one place*, rather than having to search through all of the pages to change the HTML (Web Design Group, 1997).

Style sheets offer flexibility in terms of the presentation effects that they provide. Properties such as color, background, margin, border, and many more can be applied to

all elements. With just HTML and its proprietary extensions, one must rely on attributes like **BGCOLOR**, which are only available for a few elements. Style sheets give the flexibility of applying a style to all paragraphs, or all level-two headings, or all emphasized text (Web Design Group, 1997).

However, there is a down side to using CSS. Only Netscape 4.0+, 6.0+ and Explorer 4 and 5 support CSS. Some of these browsers don't offer full support for the entire CSS1 specification (Green, 2002). For instance, if someone were to look at style-based pages in an older browser, they would not see any formatting beyond default colors, sizes, fonts and positions. However, Dreamweaver is one of the few programs that can convert styles to HTML tags automatically (Green, 2002).

Types of Style Sheets

There are several different types of style sheets, and different circumstances in which you would use them. **Embedded** style sheets are an internal part of the HTML document. All of the code is written inside the **<head>** tag of the document and affects only the one page. The following is an example of embedded CSS code:

```
<STYLE TYPE="text/css">
<!--
H1 {color:blue; font-family: verdana}
-->
</STYLE>
```

Embedded style sheets are useful when you want to apply this style to only a single page.

External style sheets are the most powerful. A single file can be used to format an infinite number of pages. Then, a single change can affect all the other pages instantly. The contents of an external style sheet file look similar to the embedded code, except that they are not part of the HTML page (Green, 2002). Instead, they are located

in a separate file, with a .css extension as opposed to .html. This .css file contains a list of styles with no other HTML code. Instead of embedding code directly into the HTML document, create an external link to the external CSS file:

```
<LINK REL=stylesheet HREF="mystyles.css" TYPE="text/css">
```

Inline style sheets are useful when you have multiple pages that share the same styles. Similar to embedded style sheets, inline styles are part of the HTML document. However, they are written in the <body> of the page, as opposed to the <head>. An example of inline code is as follows:

```
<BODY>  
<H1 STYLE="Color: orange; font-family: verdana"> Write your text here.</H1>  
</BODY>  
<HTML/>
```

Inline style sheets are not as powerful as embedded or external style sheets, primarily due to the fact that if one needed to make changes, one would have to do it every place the inline style appeared in the document. However, the advantage of using inline style sheets would be to override styles from external style sheets. Unfortunately, Dreamweaver does not automatically create code for inline styles. Therefore, it would need to be written manually.

Discussion on Box Properties

When creating your .css files, there are several categories to choose from. Type, background, block, box. borders, list, positioning and extensions are all listed in a column (See Figure 1).

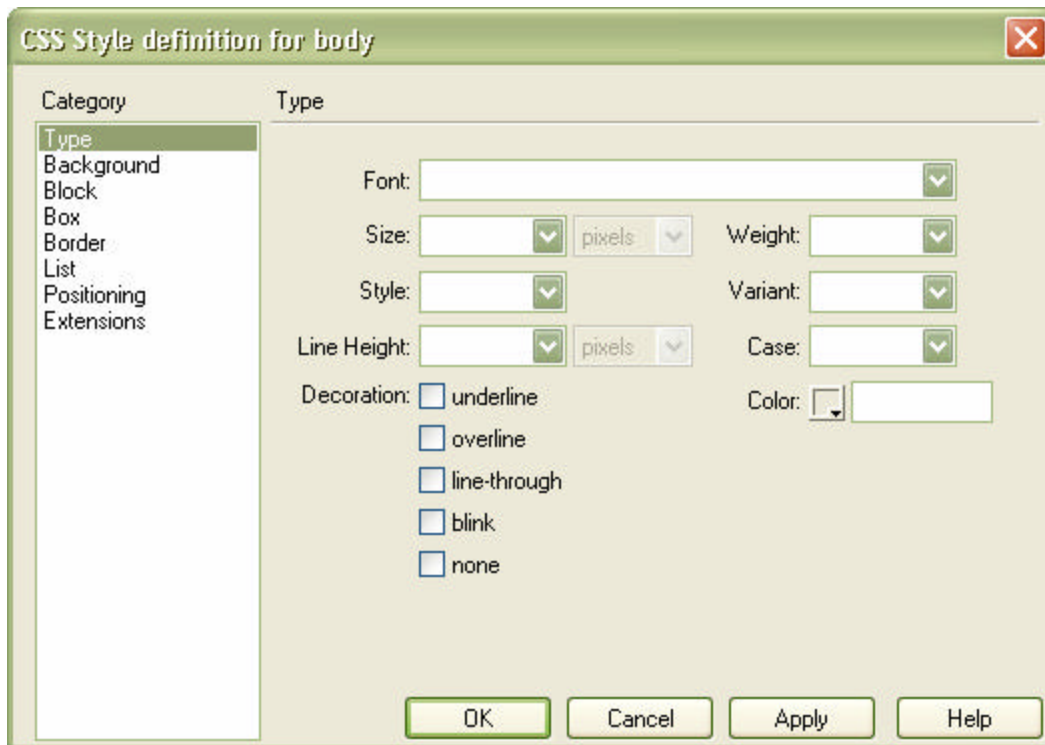
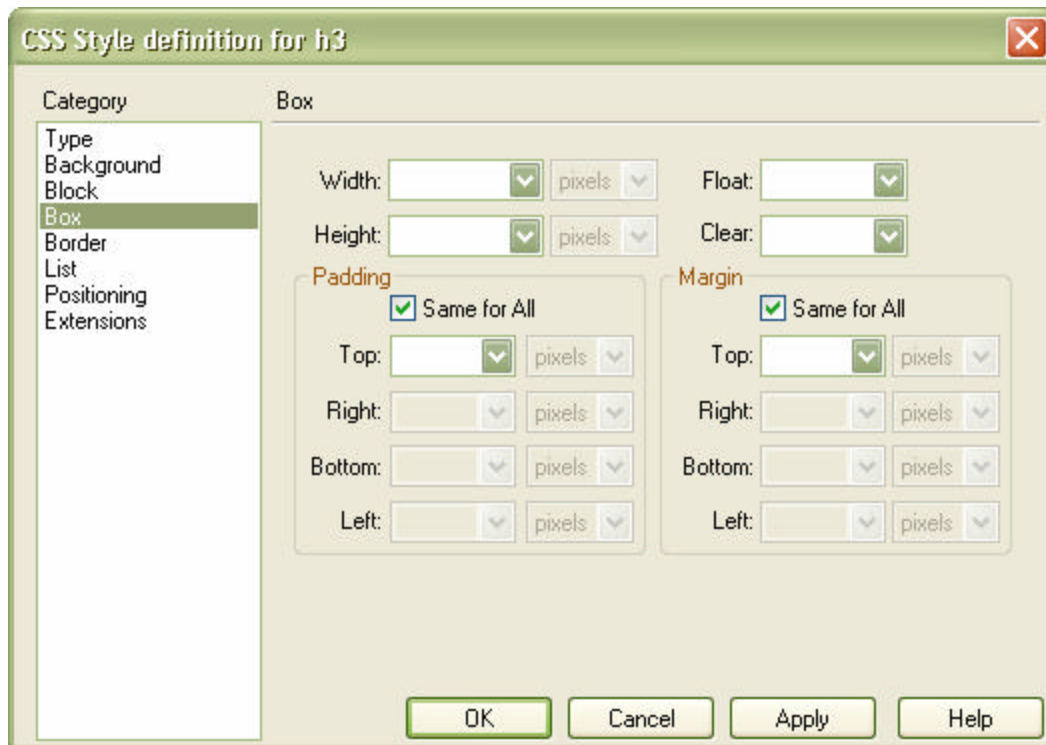


Figure 1. Categories within CSS

However, after completing Chapter 11 in the Dreamweaver 4, Hands-On Training manual, the majority of these categories were not explained. In fact, only 'Type' was introduced. In addition, since Joey Offer, a previous student in EDC385G has written a research paper on the basic principles behind cascading style sheets, I decided to explain the basic premise behind the 'Box' category.

Box Properties



Prior to discussing the box properties, it's important to have an understanding of length and height. Both *relative* and *absolute* length units are supported in CSS. Relative units give a length relative to another length property, and are preferred since they will better adjust to different media (Pozadzides & Quinn, 1997). The following relative units are available:

- **em** (ems, the height of the element's font)
- **ex** (x-height, the height of the letter "x")
- **px** (pixels, relative to the canvas resolution)

Absolute length units are highly dependent on the output medium, and so are less useful than relative units. The following absolute units are available:

- **in** (inches; 1in=2.54cm)
- **cm** (centimeters; 1cm=10mm)
- **mm** (millimeters)
- **pt** (points; 1pt=1/72in)
- **pc** (picas; 1pc=12pt)

Width

Syntax:	width: <value>
Possible Values:	<length> <percentage> auto
Initial Value:	auto
Applies to:	Block-level and replaced elements
Inherited:	No

Each block-level or replaced element can be given a width, specified as length , a percentage, or as auto (A replaced element is one for which only the intrinsic dimensions are known; HTML replaced elements include IMG, INPUT, TEXTAREA, SELECT, and OBJECT.) The initial value for the width property is auto, which results in the element's intrinsic width (*i.e.*, the width of the element itself, for example the width of an image). Percentages refer to the parent element's width. Negative values are not allowed (Pozadzides & Quinn, 1997).

This property could be used to give common widths to some INPUT elements, such as submit and reset buttons:

```
INPUT.button { width: 10em }
```

Height

Syntax:	height: <value>
Possible Values:	<length> auto

Initial Value: Auto
Applies to: Block-level and replaced elements
Inherited: No

Each block-level or replaced element can be given a height, specified as a length or as auto. (A replaced element is one for which only the intrinsic dimensions are known; HTML replaced elements include IMG, INPUT, TEXTAREA, SELECT, and OBJECT.) The initial value for the height property is auto, which results in the element's intrinsic height (*i.e.*, the height of the element itself, for example the height of an image). Negative lengths are not allowed (Pozadzides & Quinn, 1997).

.As with the width property, height can be used to scale an image:

```
IMG.foo { width: 40px; height: 40px }
```

In most cases, authors are advised to scale the image in an image editing program, since browsers will not likely scale images with high quality, and since scaling down causes the user to download an unnecessarily large file. However, scaling through the width and height properties is a useful option for user-defined style sheets in order to overcome vision problems (Pozadzides & Quinn, 1997).

Float

Syntax: float: <value>
Possible Values: left | right | none
Initial Value: None

Applies to: All elements

Inherited: No

The **float** property allows authors to wrap text around an element. This is identical in purpose to HTML 3.2's `ALIGN=left` and `ALIGN=right` for the `IMG` element, but CSS1 allows all elements to "float," not just the images and tables that HTML 3.2 allows (Pozadzides & Quinn, 1997).

Clear

Syntax: `clear: <value>`

Possible Values: `none | left | right | both`

Initial Value: None

Applies to: All elements

Inherited: No

The **clear** property specifies if an element allows floating elements to its sides. A value of `left` moves the element below any floating element on its left; `right` acts similarly for floating elements on the right. Other values are `none`, which is the initial value, and `both`, which moves the element below floating elements on both of its sides. This property is similar in function to HTML 3.2's `<BR CLEAR=left|right|all|none>`, but it can be applied to all elements (Pozadzides & Quinn, 1997).

Top Margin

Syntax: `margin-top: <value>`

Possible Values: <length> | <percentage> | auto

Initial Value: 0

Applies to: All elements

Inherited: No

The margin-top property sets the top margin of an element by specifying a length or a percentage. Percentage values refer to the parent element's width. Negative margins are permitted (Pozadzides & Quinn, 1997).

For example, the following rule would eliminate the top margin of a document:

```
BODY { margin-top: 0 }
```

Right Margin

Syntax: margin-right: <value>

Possible Values: <length> | <percentage> | auto

Initial Value: 0

Applies to: All elements

Inherited: No

The **margin-right** property sets the right margin of an element by specifying a length or a percentage. Percentage values refer to the parent element's width. Negative margins are permitted.

Example: P.narrow { margin-right: 50% }

Bottom Margin

Syntax: margin-bottom: <value>
Possible Values: <length> | <percentage> | auto
Initial Value: 0
Applies to: All elements
Inherited: No

The **margin-bottom** property sets the bottom margin of an element by specifying a length or a percentage. Percentage values refer to the parent element's width. Negative margins are permitted (Pozadzides & Quinn, 1997).

Example: DT { margin-bottom: 3em }

Left Margin

Syntax: margin-left: <value>
Possible Values: <length> | <percentage> | auto
Initial Value: 0
Applies to: All elements
Inherited: No

The **margin-left** property sets the left margin of an element by specifying a length or a percentage. Percentage values refer to the parent element's width. Negative margins are permitted (Pozadzides & Quinn, 1997).

Example:

ADDRESS { margin-left: 50% }

Note that adjoining horizontal margins are not collapsed.

Margin

Syntax: margin: <value>

Possible Values: [<length> | <percentage> | auto]{1,4}

Initial Value: Not defined

Applies to: All elements

Inherited: No

The **margin** property sets the margins of an element by specifying between one and four values, where each value is a length, a percentage, or **auto**. Percentage values refer to the parent element's width. Negative margins are permitted.

If four values are given, they apply to top, right, bottom, and left margin, respectively. If one value is given, it applies to all sides. If two or three values are given, the missing values are taken from the opposite side (Pozadzides & Quinn, 1997).

Examples of margin declarations include:

```
BODY { margin: 5em } /* all margins 5em */
```

```
P { margin: 2em 4em } /* top and bottom margins 2em,  
left and right margins 4em */
```

```
DIV { margin: 1em 2em 3em 4em } /* top margin 1em,  
right margin 2em, bottom margin 3em,  
left margin 4em */
```

Note that adjoining vertical margins are collapsed to use the maximum of the margin values. Horizontal margins are not collapsed.

Using the margin property allows one to set all margins; alternatively, the properties margin-top, margin-bottom, margin-left, and margin-right may be used.

Top Padding

Syntax:	padding-top: <value>
Possible Values:	<length> <percentage>
Initial Value:	0
Applies to:	All elements
Inherited:	No

The **padding-top** property describes how much space to put between the top border and the content of the selector. The value is either a length or a percentage. Percentage values refer to the parent element's width. Negative values are *not* permitted (Pozadzides & Quinn, 1997).

Right Padding

Syntax:	padding-right: <value>
Possible Values:	<length> <percentage>
Initial Value:	0
Applies to:	All elements
Inherited:	No

The **padding-right** property describes how much space to put between the right border and the content of the selector. The value is either a length or a percentage. Percentage values refer to the parent element's width. Negative values are *not* permitted (Pozadzides & Quinn, 1997).

Bottom Padding

Syntax: padding-bottom: <value>
Possible Values: <length> | <percentage>
Initial Value: 0
Applies to: All elements
Inherited: No

The **padding-bottom** property describes how much space to put between the bottom border and the content of the selector. The value is either a length or a percentage. Percentage values refer to the parent element's width. Negative values are *not* permitted.

Left Padding

Syntax: padding-left: <value>
Possible Values: <length> | <percentage>
Initial Value: 0
Applies to: All elements
Inherited: No

The **padding-left** property describes how much space to put between the left border and the content of the selector. The value is either a length or a percentage.

Percentage values refer to the parent element's width. Negative values are *not* permitted.

Padding

Syntax: padding: <value>

Possible Values: [<length> | <percentage>]{1,4}

Initial Value: 0

Applies to: All elements

Inherited: No

The **padding** property is a shorthand for the **padding-top**, **padding-right**, **padding-bottom**, and **padding-left** properties.

An element's **padding** is the amount of space between the border and the content of the element. Between one and four values are given, where each value is either a length or a percentage. Percentage values refer to the parent element's width.

Negative values are *not* permitted (Pozadzides & Quinn, 1997).

If four values are given, they apply to top, right, bottom, and left padding, respectively. If one value is given, it applies to all sides. If two or three values are given, the missing values are taken from the opposite side.

For example, the following rule sets the top padding to 2em, the right padding to 4em, the bottom padding to 5em, and the left padding to 4em:

```
BLOCKQUOTE { padding: 2em 4em 5em }
```

Summary

Cascading style sheets were created in order to separate content from design. If used correctly, CSS are more convenient than writing HTML code. New software, such as Dreamweaver, has made CSS more accessible for beginning web developers.

This paper is written by Amy Gottlieb for the course EDC385G Interactive Multimedia Design and Production at the University of Texas-Austin.

Resources

Bos, Bert. 2002. Cascading Style Sheets Home Page. <http://www.w3.org/Style/CSS/> Accessed October 28, 2002.

Green G. 2002. Dreamweaver 4 HOT: Hands-on Training. Berkeley, CA: Peachpit Press.

Nielson, J. 1997. *Jakob Nielson's Effective Use of Style Sheets*. <http://www.useit.com/alertbox/9707a.html> Accessed October 30, 2002.

Offer, Joey. *Cascading Style Sheets*. EDC385G Interactive Multimedia Design & Production <http://www.edb.utexas.edu/multimedia/CSS.pdf> . Accessed October 28, 2002.

Pozadzides & Quinn. 1997. CSS Properties. Web Design Group (WDG). <http://www.htmlhelp.com/reference/css/properties.html> Accessed October 31, 2002.

Pozadzides & Quinn. 1997. Cascading Style Sheets. Web Design Group (WDG). <http://www.htmlhelp.com/reference/css/> Accessed October 31, 2002.

Additional Resources

[Cascading Style Sheets:](#) Web Design Groups Cascading style sheets

[Babble List](#) Geared to advance Web Design issues, and includes a lively exchange of information, resources, theories and practices of designers and developers. It's been babbling since 1997.

[css-discuss](#) An e-mail based discussion group focused on discussing real-world uses of CSS, and helping each other to understand both how CSS is supposed to work and how browsers interpret it. The list volume tends to average between 50-100 messages per day.

[MaKo 4 CSS](#) A site that answers basic CSS questions. Also, the site has some nice information on handling Netscape Navigator 4 issues.

[House of Style](#) A collection of resources for web developers who want to work with Cascading Style Sheets and other web standards such as HTML 4.01 and XHTML 1.0.

[Style Sheets Reference Guide](#) [css/edge](#) Intended, first and foremost, to be creative with CSS. It does not exist to present or explain safe cross-browser techniques; in fact, almost the opposite. The goal here is to find ways to make CSS live up to its fullest potential, with only minimal regard to browser limitations.

[websitetips.com CSS section](#) Additional links to CSS resources

[Glish CSS Layout Techniques](#) A resource of Web page layouts using CSS.

[Blue Robot Layout Reservoir](#) Another good site that shows CSS layouts by example.

[Noodle Incident Little boxes](#) Another collection of two and three column CSS-enabled layouts.

[CSS Pointers Group](#) A site focused on compiling a useful source of information, examples and links to other external resources

[CSS2 Specification](#) The specification documentation that defines Cascading Style Sheets, level 2 (CSS2).

[CSS2 Validator](#) A free service that checks your CSS for conformance to W3C Recommendations and other standards.

[\(X\)HTML Validator](#) A free service that checks your documents like HTML and XHTML for conformance to W3C Recommendations and other standards.

[W3C "Core" Stylesheets](#) A site that offers authors an easy way to start using style sheets without becoming designers

[Web Standards Project](#) A grassroots coalition fighting for standards that ensure simple, affordable access to web technologies for all.

[Advanced CSS layouts: Step by Step](#) Replicate the webreference. com home page using CSS for layout,

[Everything you ever wanted to know about Style](#) Very thorough account on CSS

[From Hacks to Standards: A Web Designer's Journey](#) — Jeffrey Zeldman's report from the CSS front. Describes the battle he waged while trying to update [A List Apart](#) site with

a CSS layout. There is more good information in that article than is dreamt of in your philosophy. And there is some follow up info [here](#).

[Box Lessons](#) — Owen Briggs has expanded his tutorial into a very nice resource that not only provides some very useful layouts, but also details all the aggravating problems he encountered while developing them.

[Design-o-rama](#) at glassdog.com —a very useful tutorial for getting started with CSS.