ClickStart
GUIDES

CSS
TO THE
POINT

Quick solutions
to all of your
CSS questions —
comprehensive
and to the point!

by **Scott DeLoach**

CSS *to the point*

Scott DeLoach

Contents

Lists   49

Links   61

Images 71

Tables 79

Forms   93

Print   99

Page layout   105

Testing 123

Resources 129

CSS Quick Reference 131

Index 146

Introduction

This book was designed to help you find quick solutions to common CSS questions. The sections use a question-answer format, with a short description of the question, a visual example, and the CSS solution.

All of the solutions have been tested in Internet Explorer, Firefox, Opera, and Safari. Most of the CSS solutions are supported by the four major browsers, and notes are included when there is not a cross-browser solution.

Icons used in this book

The following icons are used throughout this book to help you find important and time-saving information.

Icon	Meaning	Description
⚠	Caution	Information about an unsupported CSS property.
◇	Note	Additional information about a topic.
TIP▶	Tip	A recommended best practice, shortcut, or workaround.

Updates

For the most up-to-date information about this book, see www.clickstart.net.

Overview

This section provides a focused, "to the point" overview of CSS.

What does "CSS" stand for?

"CSS" stands for "Cascading Style Sheets."

The word "cascading" is used because CSS formatting flows (or "cascades") from general rules to specific rules. For example, you can format the body tag to use black text. This rule will flow down to everything in your document, so you don't have to format each element (such headings, paragraphs, lists, and tables) individually to use black text. Each element will automatically inherit the black text setting from the body tag.

What is CSS?

Cascading style sheets can be used to format HTML, XHMTL, and XML documents. CSS is useful because it allows you to separate your content from its appearance. For example, paragraphs are contained within p (paragraph) tags, but CSS style rules specify how paragraphs are formatted. With CSS, a document's formatting can be easily adjusted for reading online or on paper, for viewing or listening (for hearing-impaired users), or for viewing on a PC or PDA. CSS even allows readers to adjust the text size if it's too small to read.

Who created CSS?

CSS was created by Bert Bos and Håkon Wium Lie. It became a W3C ("World Wide Web Consortium") recommendation in 1996.

Where can I view the full CSS standard?

The W3C prefers to use the term "recommendation" rather than "standard," but many people refer to their specifications as standards anyway. You can view and download the CSS recommendation at:

www.w3.org/Style/CSS

Where do I add my styles?

You can create inline, embedded, and external styles. Each of these options is described below, but here's a quick overview:

Type	Where it's located
Inline	Inside a tag
Embedded	In the <head> of a topic as a <style>…</style> block
External	In a separate css file

What are inline styles?

Inline styles are placed inside a tag:

```
<p style="font-family: Arial; color: red;">This text would be red.</p>
```

In short: inline styles are bad. They are hard to update, inefficient, and add unneeded code to your documents. For example, if you needed to change the paragraph above to blue text, you would have to find it in your document to change it to blue. That's not too hard for one paragraph, but it would be tedious to change 10 paragraphs. Imagine having to change 100 paragraphs in multiple documents. No thanks!

Another problem with inline styles is that you have to remember the formatting. For example, if you want to format paragraphs in multiple documents as 12pt red Arial, you have to remember those values. That can be hard to do with multiple types of content: headings, body text, notes, tables, bulleted lists, numbered lists, etc.

What are embedded styles?

Embedded styles are placed in the <head> section of a document:

```
<head>
<style>
p { font-family: Arial; }
</style>
</head>
```

This embedded style rule would format all paragraphs in the document to use the Arial font family.

The downside of embedded styles is that you cannot reuse them in multiple documents. You have to include the style information in each document, and you have to update each document if you change the styles. If you only have one document, you might as well use embedded styles. That way, all of your formatting and your content are in the same file. However, you probably have more than one document to format. That's when external styles become really useful.

What are external styles?

External styles are placed in a separate text document with a css extension. You can include multiple style rules in one external style sheet, and you can link multiple documents to the same external style sheet.

How do I link a document to a style sheet?

You can link a document to a style sheet using the **link** element:

```
<html>
 <head>
  <link rel="stylesheet" type="text/css" href="styles.css" />
 </head>
 <body>
  <p>Styles are cool!</p>
 </body>
</html>
```

How do I write a style rule?

A CSS rule begins with a selector identifying a target element. The selector is followed by a declaration block listing one or more properties of that element and a value to be applied to each property.

```
h1 { color: black; }
```

In this example, "h1" is the selector, "color: black" is a declaration, "color" is a property, and "black" is a value.

If you want to apply the same formatting to multiple selectors, you can separate the selectors with a comma. The following example applies the same formatting to h1s (heading 1) and h2s (heading 2):

```
h1, h2 { color: black; background: yellow; }
```

How can I easily format all of my content?

You can use the * (the "wildcard" or "universal") selector to format everything in a document:

```
* { color: red; }
```

Another approach is to use the body tag. A basic HTML document could be written as:

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="styles.css" />
  </head>
  <body>
    <h1>Sample heading</h1>
    <p>Styles are cool!</p>
  </body>
</html>
```

An XHTML or HTML document contains two main sections: the head and the body. The * selector includes both sections, but the head section only contains "set up" information. There's no reason to format it because the user doesn't view it. So, the following style rules are basically the same:

```
* { color: red; }
body { color: red; }
```

How do I format a specific paragraph?

You can use a class or ID to format a specific block of content, such as a paragraph, list item, or table cell.

Classes – as many as you need

If you need to format multiple (but not all) paragraphs, you should use a class. For example, you might need to format three "note" paragraphs as red text. First, you would need to assign the class to each note:

```
<p class="note">This is a note.</p>
```

Then, you could write a style rule to format your notes:

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

This note class will first format a block of content as a paragraph and then change the text to red. If paragraphs are formatted as "Arial," your notes will be Arial and red:

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

If paragraphs are formatted as black text and your note class specifies red text, the note class will change your notes to red. Other paragraphs will still use black text.

IDs – one per document

You can only use an ID once in a document. You might use an ID to identify a copyright statement:

```
<p id="copyright">Copyright 2008</p>
```

You can then format the ID in your style sheet:

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

Generic classes and IDs

The class and ID examples above are both based on the p (paragraph) tag. You can create a class or ID for any element, such as heading 1s (h1), tables, or lists:

```
h1.pageTitle { color: red; }
```

However, classes or IDs that are created for a specific tag cannot be used for another tag. For example, the "p.note" class cannot be applied to a heading—It can only be applied to a paragraph.

You can create a generic class or ID that can be applied to any tag by not including a tag in your style rule:

```
.note { color: red; }
#copyright { color: red; }
```

This generic "note" class or generic "copyright" ID could be applied to any tag, such as h1, p, li (list item), or td (table data).

How do I format a specific word or character?

You can use a span to format a word or character. For example, to boldface the word "Start" in "Click**Start**," you would first need to add a span:

```
<p>For details, see the Click<span>Start</span> website.</p>
```

You could format all spans to be bold:

```
span { font-weight: bold; }
```

Or, you could use a class to boldface only certain spans:

CSS rule
```
span.emphasized { font-weight: bold; }
```

How do I specify colors?

You can specify colors using rgb ("red-green-blue") values, hexadecimal values, or keywords.

All of these style rules would set the h1 tag to use red text:

```
h1 { color: red; }
h1 { color: rgb(100%, 0%, 0%); }
h1 { color: rgb(255, 0, 0); }
h1 { color: #ff0000; }
h1 { color: #f00; }
```

For a list of CSS color keywords, see:

www.w3.org/TR/CSS21/syndata.html#color-units

TIP *If a color's hexadecimal value uses pairs of numbers, like #aabbcc, you can use CSS shorthand to write the value as #abc.*

What happens if multiple rules apply to a block of content?

If multiple rules apply to a block of content, CSS uses "specificity" rules to determine which rule wins.

Inline styles override embedded styles, and embedded styles override external styles. For example, if a paragraph's inline styles specify green text, the text will be green no matter what color is specified in an embedded or external style rule.

IDs also override classes. For example:

```
<p class="sampleClass" id="sampleID">text text text</p>
```

First, the p tag's style rule is applied:

```
p {
  color: black;
  background-color: green;
}
```

Next, the class style rule is applied:

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

Finally, the ID style rule is applied:

```
p#sampleID { color: yellow; }
```

The final formatting would be yellow (from the ID rule) Arial (from the class rule) text with a green background (from the p tag).

How do I override inline styles?

You can use the **!important** keyword to override inline styles. For example, you might have the following inline formatting in a document:

```
<p style="color: red; background-color: yellow;">text </p>
```

The following style rule will override the inline formatting in the example above:

```
p {
  color: black !important;
  background-color: white !important;
}
```

You must include the !important keyword for each property you want to override.

How should I order my rules in my style sheet?

You can arrange your style rules in any order.

Most CSS authors begin with rules for higher-level tags, such as * or body, and then rules for elements in the order they appear on the page: headings first, then paragraphs, tables, and lists. I often place my link styles near the top before the heading rules.

How do I add comments to my style sheet?

You can add comments to your style sheet between /* and */:

```
/* This is a one-line comment */

/*
  This is a
  multi-line
  comment
*/
```

Comments often include the author's name, last modified date, color definitions (e.g., "#aabbff is the light blue in our company logo"), and notes about browser issues.