

Steve's Tech Resource

The Web Development, Internet, Software, Hardware, and Multimedia Resource

[HOME](#) [ABOUT](#) [WHAT'S NEW](#) [LOGIN](#) [JOIN](#) [SEARCH](#) [BY DATE](#) [ALL BOARDS](#)

On CSS 294KB

Published: 29Jun01 | Last Updated: 10Jun11 | Status: To Be Continued

1. [Introduction](#)
2. [Definitions](#)
3. [CSS Basics](#)
 - 3.1. [CSS Comments](#)
 - 3.2. [Grouping Selectors And Grouping Declarations](#)
 - 3.3. [Valid Properties, Values, And The Handling Of Invalid Styles](#)
 - 3.4. [The HTML/XHTML Document Tree And The Inheritance Of Style](#)
 - 3.5. [Important Styles](#)
 - 3.6. [Style Rule Syntax And Conventions](#)
4. [Creating Style Rules And Attaching Styles To Content](#)
 - 4.1. [Elements As Selectors](#)
 - 4.1.1. [The Element, Body, As Selector And Defining A Default Style For A Web Page/Site](#)
 - 4.1.2. [Elements Other Than The Element, Body, As Selectors](#)
 - 4.2. [ClassNames As Selectors](#)
 - 4.2.1. [Classing The Generic Inline Element, Span, And The Generic Block-Level Element, Div](#)
 - 4.3. [Pseudo-Classes As Selectors](#)
 - 4.4. [Inline Styles](#)
5. [Style Sheets](#)
 - 5.1. [Embedded Style Sheets](#)
 - 5.2. [External Style Sheets](#)
 - 5.3. [The Advantage Of External Style Sheets Over Embedded Style Sheets](#)
 - 5.4. [Imported Style Sheets](#)
6. [User Agent Default Styles, Author Styles, And User Styles](#)
 - 6.1. [User Agent Default Styles](#)
 - 6.2. [Author Styles](#)
 - 6.3. [User Styles](#)
7. [The Cascade](#)
 - 7.1. [Overview Of The Cascade](#)
 - 7.1.1. [The Cascade By Diagram: Author Styles Present. User Styles Present.](#)
 - 7.1.2. [The Cascade By Diagram: Author Styles Present. User Styles Absent.](#)
 - 7.1.3. [The Cascade By Diagram: Author Styles Absent. User Styles Present.](#)
 - 7.1.4. [The Cascade By Diagram: Author Styles Absent. User Styles Absent.](#)
 - 7.2. [The Cascade Processing Author Styles Into An Author Styles List And User Styles Into A User Styles List](#)
 - 7.3. [The Cascade Merging The User Agent Default Styles, Author Styles List, User Styles List, And The Definition Of A Style Conflict](#)
 - 7.3.1. [The Definition Of A Style Conflict: Styles With The Same Property, The Same Value, But Different Selectors. No Style Conflict.](#)
 - 7.3.2. [The Definition Of A Style Conflict: Styles With The Same Selector, But Different Properties. No Style Conflict.](#)
 - 7.3.3. [The Definition Of A Style Conflict: Styles With The Same Selector, The Same Property, And The Same Value. No Style Conflict.](#)
 - 7.3.4. [The Definition Of A Style Conflict: Styles With The Same Selector, The Same Property, But Different Values. Style Conflict.](#)
 - 7.4. [The Cascade's Rules For Resolving Style Conflicts](#)
 - 7.5. [The Firefox 3.5 Cascade In Action](#)
 - 7.5.1. [Approximating The Firefox 3.5 User Agent Default Styles](#)
 - 7.5.2. [The Firefox 3.5 Cascade Processing Author Styles Into An Author Styles List](#)
 - 7.5.3. [The Firefox 3.5 Cascade Merging The Firefox 3.5 User Agent Default Styles, Author Styles List, And Resolving Style](#)

Conflicts

7.5.4. [The Firefox 3.5 Cascade Creating The Master Style Sheet](#)

7.5.5. [The Firefox 3.5 Rendering Engine Applying The Styles Of The Master Style Sheet To Content: Proof That This Is The Way CSS Works](#)

8. Additional Reading

1. Introduction

Important Note: Some of the concepts/definitions in this [Web page](#) are specific to [content](#) and, therefore, are not intended to have a universal meaning.

HTML/XHTML is a [Web language](#) in which [markup](#) is used to attach two types of [structural information](#) to [content](#): 1.) a structure type (e.g., heading, paragraph, etc.), and 2.) the demarcation of the starting and ending points for the application of the structure type.

HTML/XHTML provides numerous [elements](#) for structuring content. The structure type associated with each element is unique and is intended for a specific purpose. A couple elements are listed below:

Element	Name	Markup	Structure Type/Intended Purpose
h1	heading 1	<h1></h1>	A single line of text as a block , large font, bold, margin top and bottom.
p	paragraph	<p></p>	One or more sentences as a block, normal font, margin top and bottom.
table	table	<table> </table>	Content as a block organized into columns and rows through the use of child elements , th, tr, and td.
ol	ordered list		Content as a block organized into a numbered (i.e., ordered) list through the use of child element, li.

In the mid 1990's, a [standard](#) for attaching [style](#) to content did not exist and [HTML](#) was in its infancy. As a result, [Web pages/sites](#) looked rather similar and unattractive and the future direction of style and HTML were open for discussion.

Also in the mid 1990's, a fierce competition for dominance in the emerging [Web Browser](#) market developed between Netscape and Microsoft and their respective Web Browsers, Netscape Navigator and Microsoft Internet Explorer. Known as the Web Browser war, its most conspicuous and polarizing battle centered around claims alleging Microsoft of monopolistic business practices with respect to bundling/integrating Microsoft Internet Explorer with Microsoft Windows 95 and Windows 98.

While most were preoccupied picking and defending sides in the impending antitrust lawsuit, another battle in the Web Browser war drew relatively little attention: Netscape and Microsoft, apparently sensing a void and an opportunity not only to appease the desire of [authors](#) and [users](#) for style, but to influence the future direction of style and HTML, started creating elements and [attributes](#) related to style and releasing versions of their Web Browsers that supported their own more or less proprietary versions of HTML.

Those with a nonpartisan interest in the future direction of style and HTML, however, recognized that the "silent" battle in the Web Browser War was creating three serious problems: 1.) HTML was morphing into a structure/style combination language; 2.) HTML was splitting into two distinct, Web Browser-specific languages; and 3.) style, instead of being a positive for authors, was becoming a negative for it had less to do with creating attractive Web pages/sites than keeping track of the growing differences between Netscape Navigator and Microsoft Internet Explorer with respect to style.

Tim Berners-Lee, who invented the [WWW](#) in 1989 and wrote the first Web Browser and Web Server in 1990, along with others, founded the W3C in 1994. The W3C's mission is: "To lead the [World Wide Web](#) to its full potential by developing protocols and guidelines that ensure long-term growth for the [Web](#)." With respect to style and HTML, the W3C published the following [Recommendations](#) between December 1996 and May 2001:

Date	Recommendation	Significance
Dec 1996	CSS1	The first W3C CSS Recommendation .
Jan 1997	HTML 3.2	The first W3C HTML Recommendation: <ul style="list-style-type: none"> • A major update to Tim Berners-Lee's HTML 2.0 Request For Comment published in November 1995.

		<ul style="list-style-type: none"> • Did not support CSS, but hinted at adding CSS support in future HTML Recommendations.
Dec 1997	HTML 4.0	<p>A major update to the HTML 3.2 Recommendation:</p> <ul style="list-style-type: none"> • CSS support added. • <u>Deprecated elements and attributes</u> related to <u>style</u>. • Included a Transitional DTD and a Strict DTD, which included and excluded, respectively, elements and attributes related to style that the W3C expected to phase out as being replaced by CSS.
Apr 1998	HTML 4.0, Revised	Primarily a document update that corrected errata in the December 1997 HTML 4.0 Recommendation.
May 1998	CSS2	A minor update to the CSS1 Recommendation.
Dec 1999	HTML 4.01	A minor update to the HTML 4.0 Recommendation.
Jan 2000	XHTML 1.0	<p>A reformulation of HTML 4 as an XML 1.0 application:</p> <ul style="list-style-type: none"> • A major update to the HTML 4.01 Recommendation. • Like the HTML 4.01 Recommendation, deprecated elements and attributes related to style. • Like the HTML 4.01 Recommendation, included a Transitional DTD and a Strict DTD, which included and excluded, respectively, elements and attributes related to style that the W3C expected to phase out as being replaced by CSS.
May 2001	XHTML 1.1	<p>A reformulation of XHTML 1.0 Strict using XHTML modules:</p> <ul style="list-style-type: none"> • A major update to the XHTML 1.0 Recommendation. • <u>Obsoleted</u> elements and attributes related to style. • Eliminated the XHTML 1.0 Recommendation Transitional DTD. • Included a single DTD similar to the XHTML 1.0 Recommendation Strict DTD, but built on XHTML modules.

Concerning style, the W3C described a new Web language, **CSS**, in which: 1.) style is attached to content; 2.) the attachment of style to content is separate from the attachment of structure to content (i.e., although CSS and HTML act upon the same content, they are distinct Web languages); and 3.) the application of style to content Cascades.

Concerning HTML/XHTML, the W3C Recommendations accomplished the following; 1.) CSS support was added, and 2.) elements and attributes related to style were first deprecated and then obsoleted from HTML/XHTML as being replaced by CSS.

Note:

- The W3C does not have the authority to impose the implementation of its Recommendations. The W3C does, however, allow parties interested in the W3C Recommendations to become W3C members, which allows them to contribute to the W3C Recommendations. It is in allowing W3C members to contribute to the W3C Recommendations that the W3C Recommendations acquire their weight.
- A couple of the styles created by Netscape and Microsoft were incorporated into the W3C Recommendations. The others were discarded and became known as nonstandard HTML extensions.

The W3C CSS1 and HTML 4.0 Recommendations provided a solution to the aforementioned problems arising from Netscape's and Microsoft's attempts not only to appease the desire of authors and users for style, but to influence the future direction of style and HTML. For the solution to become a reality, however, not only would authors need to implement the W3C CSS1 and HTML 4.0 Recommendations into their Web pages/sites, but Web Browser vendors would need to implement the W3C CSS1 and HTML 4.0 Recommendations into their Web Browsers.

Authors applauded the W3C CSS1 and HTML 4.0 Recommendations; 1.) the distinction between style and structure made sense, 2.) most of the styles authors desired were accounted for, 3.) they were easy to learn and implement, and, 4.) most importantly, being standards, they promised an end to the growing differences between Netscape Navigator and Microsoft Internet Explorer with respect to style.

At the start of 1998, Microsoft Internet Explorer's support for the W3C CSS1 and HTML 4.0 Recommendations, albeit imperfect and partial, was superior to that of Netscape Navigator. But far more importantly, that Netscape Navigator's and Microsoft Internet Explorer's support for the W3C CSS1 and HTML 4.0 Recommendations differed only served to increase, by at least one order of magnitude, the differences between Netscape Navigator and Microsoft Internet Explorer with respect to style. As a result, authors eager to design Cross-Browser compatible Web pages/sites implementing the W3C Recommendations anxiously awaited the next releases of Netscape Navigator to see if its support for the W3C Recommendations was closing the gap with Microsoft Internet Explorer.

In February 1998, in apparent acknowledgement that unless something drastic happened Microsoft was going to win the Web Browser war, Netscape formed the Mozilla Organization. The Mozilla Organization created the Mozilla Project whose goal was to develop community driven, open source, cross-platform software for the WWW, including a standards conforming Web Browser. Authors and users, particularly those intrigued by an open source alternative to Microsoft's proprietary software, welcomed this news. But when the Mozilla Project announced that their new Web Browser would have to be developed from scratch, and that, as a result, it would take considerably longer than initially anticipated for the new Web Browser to be released, all eyes refocused on the next releases of Netscape Navigator.

In May 1998, the W3C published its second CSS Recommendation, CSS2. Five months later, in October 1998, Netscape released Netscape Navigator 4.5. When it was determined that Netscape Navigator 4.5's support for the dated W3C CSS1 Recommendation was only slightly better than that of Netscape Navigator 4.0, it started to sink in that Netscape Navigator's poor support for the W3C CSS Recommendations could be because its code base was not amenable to change.

In March 1999, Microsoft released Microsoft Internet Explorer 5.0. Microsoft Internet Explorer 5.0's vastly improved support for the W3C CSS Recommendations increased, by at least another level of magnitude, the differences between Netscape Navigator and Microsoft Internet Explorer with respect to style. The headaches this caused authors eager to design Cross-Browser compatible Web pages/sites implementing the W3C Recommendations cannot be overstated: Web pages/sites that appeared more or less as expected in Microsoft Internet Explorer 5.0 could very well appear completely discombobulated in Netscape Navigator 4.5. In fact, the gap between Netscape Navigator and Microsoft Internet Explorer with respect to style was now so large that many authors resorted to adding "Best Viewed With Web Browser X" logos to their Web pages and/or using CSS and HTML "hacks" to trick Web Browsers into displaying their Web pages a particular way. Moreover, and as it became apparent that the antitrust lawsuit against Microsoft was not going to drastically affect Microsoft Internet Explorer's newfound leading position in the Web Browser market, innumerable authors, including those who were pro-Netscape and/or anti-Microsoft, not only started denouncing Netscape Navigator, but anxiously awaited its market share to drop to low single digits so they could design Web pages/sites implementing the W3C Recommendations without having to worry about Netscape Navigator compatibility.

Numerous factors played a role in Netscape Navigator's defeat to Microsoft Internet Explorer in the Web Browser war, and, therefore, what role Netscape Navigator's poor support for the W3C Recommendations played in its demise is debatable. One thing, however, is not debatable: the authoring community let it be known that if a Web Browser vendor wanted good standing with the authoring community, it better keep up with the competition in supporting the W3C Recommendations.

In December 2000, after over two years of development, the Mozilla Project released their Web Browser, named Mozilla. As promised, Mozilla's support for the W3C Recommendations was comparable to that of Microsoft Internet Explorer and authors and users finally had a standards conforming Netscape affiliated alternative to Microsoft Internet Explorer.

Note: Mozilla was never intended as an "official" product, but simply as vehicle for distributing and testing the Mozilla Project's new rendering engine, Gecko, as well as other features to be incorporated into the official product, Netscape, which, for various reasons, never gained much popularity.

Although Mozilla and Netscape are no longer being developed, the Mozilla Project continues to develop Gecko and to build it into a number of so-called Mozilla-based Web Browsers. The Mozilla-based Web Browsers currently being developed including Firefox, Flock, and SeaMonkey.

2. Definitions

Important Note: Some of the concepts/definitions in this Web page are specific to content and, therefore, are not intended to have a universal meaning.

ancestor

- An element in the same branch as, and one or more levels above, another element in the HTML/XHTML document tree. Compare with descendant.
- An element's parent, its parent's parent, its parent's parent's parent, etc.

attribute

- A characteristic of content defined by HTML/XHTML.
- Those related to style, except for the attributes, class and style, are deprecated or obsoleted from HTML/XHTML as being replaced by CSS.
- Syntax: <element attribute="value">content</element>.
- Examples: face, color, size, class, and style.
- A generic terms referring to an attribute and value pair.

author

- A person who writes (i.e., [authors](#)) [Web documents](#). Compare with [user](#) and [user agent](#).
- One of the three sources of [style](#). See [author styles](#).

author styles

- [Styles](#) written by an [author](#). Compare with [user styles](#) and [user agent default styles](#).
- Includes [embedded style sheets](#), [external style sheets](#), [imported style sheets](#), and [inline styles](#).

author styles list

- The list of [author style rules](#) associated with a [Web page](#), replacing any links to [external style sheets](#) with the [external style sheets](#), themselves, and replacing any `@import` statements with the [external style sheets](#) to be imported, themselves. Compare with [user styles list](#)
- Created by the [Cascade](#).

block-level element

- An [element](#) type in which [content](#) is physically separated (i.e., blocked off) from adjacent [content](#) as if preceded and followed by a line break. Compare with [inline element](#).
- Examples: `h1`, `p`, `ol`, `ul`, `table`, `form`, and `div`.

body

- The [element](#) that is the root (i.e., origin) and [ancestor](#) of [content](#).

cascade

- *"Something arranged or occurring in a series or in a succession of stages so that each stage derives from or acts upon the product of the preceding."* - Merriam-Webster's Collegiate Dictionary, Tenth Edition, 2001.
- A component of a [user agent](#).
- For each [Web page](#): 1.) processes [author styles](#), if present, into an [author styles list](#), and [user styles](#), if present, into a [user styles list](#); 2.) merges the [user agent default styles](#), [author styles list](#), if present, and [user styles list](#), if present; 3.) resolves [style conflicts](#); and 4.) creates a master [style sheet](#) whose [styles](#) the [user agent's rendering engine](#) applies to [content](#) in the [Web Browser window](#).
- Also: 1.) allows [user agent default styles](#), [author styles](#), and [user styles](#) to simultaneously [style content](#); 2.) allows [author styles](#) and [user styles](#) to be partial [styles](#) (i.e., [author styles](#) and [user styles](#) need only assign the [styles](#) to add to, or change from, the [user agents default styles](#)); 3.) provides modularity (i.e., [authors](#), instead of needing to create a single large [style sheet](#) encompassing an entire [Web page/site](#), can think of their [Web pages/sites](#) as consisting of sections (i.e., modules) each getting its own module-specific [style sheet](#)); and 4.) provides [author/user](#) balance (i.e., a mechanism for [author styles](#) to override [user styles](#) and, conversely, for [user styles](#) to override [author styles](#)).

child

- An [element](#) in the same branch as, and one level below, another [element](#) in the [HTML/XHTML document tree](#).
- An [element](#) that is contained within (i.e., enclosed by) another [element](#). Compare with [parent](#).
- The [descendant](#) that is only one level below another [element](#) in the [HTML/XHTML document tree](#).

class

- The [attribute](#) for attaching [style](#) to [content](#) via the [selector](#), [className](#). Compare with [style](#), [HTML](#).
- An [attribute](#) added to the [W3C HTML 4.0 Recommendation](#) as part of its support for the [W3C CSS1 Recommendation](#).
- A generic term referring to the [attribute](#), [class](#), and/or the [selector](#), [className](#).

classed element

- An element with the [attribute](#), [class](#), assigned value, [className](#), in [markup](#).
- Syntax: `<element class="className">content</element>`.

className

- An [author-defined](#) variable that functions as a [selector](#). Compare with [element](#) and [pseudo-class](#).
- The [value](#) assigned to the [attribute](#), [class](#).

Note: The attribute, class, was added to the W3C HTML 4.0 Recommendation as part of its support for the W3C CSS1 Recommendation. The attribute, class, behaves different than other HTML/XHTML attributes in that the value assigned to the attribute, class, declares a variable whose name is the value assigned to the attribute, class, and whose value is a declaration block. That the value assigned to the attribute, class, behaves like a variable name is why className, rather than, for example, classValue, is used in this Web page to represent the value assigned to the attribute, class.

code

- See source code.
- To write source code.

comment, CSS

- Text in a style sheet that is not interpreted by, and, therefore, is ignored by, a user agent.
- Typically serve as a note/reminder (i.e., comment) about the surrounding, interpreted code.
- Preceded by `/*` and followed by `*/` characters, no quotes.
- Are single line; meaning, they can wrap, but cannot span multiple lines.
- Syntax:
selector { declaration block } /* CSS comment at the end of a line. */
/* CSS comment at the start of a line. */

content

- The text, links, images, etc. that appear in the Web Browser window upon which HTML/XHTML structure and CSS style is applied.
- That between an HTML/XHTML document's `<body></body>` tags apart from the markup.

Cross-Browser compatibility

- That which allows Web pages to appear and function identically, or without significant difference, in various Web Browsers.
- Related to; 1.) the capabilities of various Web Browsers, 2.) the author's knowledge of the capabilities of various Web Browsers, and 3.) the author's ability to write HTML/XHTML documents and style sheets that appear and function identically, or without significant difference, in various Web Browsers.

CSS

- Cascading Style Sheets.
- The Web language in which style is attached to content.

CSS document

- See external style sheet.

declaration

- A style.
- Consists of a property and value pair. Compare with attribute.
- Syntax: selector { propertyOne: valueOne; propertyTwo: valueTwo; propertyThree: valueThree } [a style rule with three declarations (i.e., styles)].

declaration block

- One of the two parts of a style rule. Compare with selector.
- Assigns one or more styles to a selector.
- Consists of one or more declarations, each declaration representing a style.
- Syntax: selector { declarationOne; declarationTwo; declarationThree } [a declaration block with three declarations].

deprecated

- Elements and attributes defined by a W3C HTML/XHTML Recommendation that are designated to be obsoleted from future W3C HTML/XHTML Recommendations as being replaced by CSS. Compare with obsolete.
- Elements and attributes that user agents are encouraged to support until they become obsoleted.

descendant

- An element in the same branch as, and one or more levels below, another element in the HTML/XHTML document tree. Compare with ancestor.
- An element's children, its children's children, its children's children's children, etc.

DTD

- Document Type Definition.
- The part of a W3C HTML/XHTML Recommendation that defines the language's proper usage, including, but not limited to, the valid elements, attributes, and value types.

dynamic pseudo-class

- The pseudo-class type for attaching style to content (which, by default, includes links) based on the content's state with respect to user interaction with the content. Compare with link pseudo-class.
- States include: 1.) :hover, for attaching style to content with the mouse cursor placed over it; 2.) :active, for attaching style to content between the mouse down click and the mouse up click; and 3.) :focus, for attaching style to content ready to accept input.

element

- An HTML/XHTML structure type attached to content through the use of markup.
- A selector. Compare with className and pseudo-class.
- Divided into two types; 1.) block-level elements, and 2.) inline elements.
- Examples: body, h1, p, ol, ul, table, form, div, a, em, i, strong, b, img, and span.

embedded style sheet

- A.k.a., inline style sheet.
- A style sheet located inside (i.e., embedded within) an HTML/XHTML document. Compare with external style sheet and imported style sheet.
- Style rules inside `<style type="text/css"></style>` tags placed in the `<head></head>` section of an HTML/XHTML document.
- Written by authors.

external style sheet

- A style sheet located outside (i.e., external to) an HTML/XHTML document. Compare with embedded style sheet and imported style sheet.
- Style rules in a text file having a .css file extension.
- A style sheet associated with (i.e., linked to) an HTML/XHTML document by adding the line, `<link rel="stylesheet" type="text/css" href="anyFileName.css" />`, where the value assigned to the attribute, href, is an absolute or relative path to an external style sheet, anyFileName.css, to the `<head></head>` section of an HTML/XHTML document. Written by authors.
- A style sheet attached to content via a user agent's Options/Preferences. Written by users.

HTML

- HyperText Markup Language.
- The primary publishing language of the WWW.
- The Web language in which markup is used to attach two types of structural information to content: 1.) a structure type (e.g., heading, paragraph, etc.), and 2.) the demarcation of the starting and ending points for the application of the structure type.

HTML/XHTML

- HTML and/or XHTML.

HTML/XHTML document

- A text file written in HTML or XHTML having a .htm or .html file extension.

HTML/XHTML document tree

- A.k.a., document tree.
- A diagram of the elements that constitute an HTML/XHTML document. The diagram is similar in shape to a tree in that it has a root and branches. By convention, though, the root is placed at the top of the diagram.
- Depicts and is frequently used to define the terms, parent, child, sibling, ancestor, descendant, and inheritance.

imported style sheet

- An external style sheet inserted (i.e., imported) into an embedded style sheet, external style sheet, or imported style sheet. Compare with embedded style sheet and external style sheet.
- Written by authors and users.

inheritance

- The transferring of style from ancestors to descendants and the receiving of style by descendants from ancestors.

inline element

Note: Some inline elements (e.g., em, i, strong, and b) are more related to style than structure, which contradicts the definition of an element as an HTML/XHTML structure type. It makes sense, though, that the W3C has retained these old, stylistically simple, frequently used, and easy to deploy elements instead of obsoleting them as being replaced by CSS.

- A.k.a., text-level element.
- An element type in which content remains besides (i.e., inline with) adjacent content. Compare with block-level element.
- Examples: a, em, i, strong, b, img, and span.

inline style

- A style, style rule and style sheet independent, attached to content via the attribute, style, assigned a value whose syntax mimics that of a declaration block, minus the braces.
- Syntax: <element style="property: value">content</element>.

Internet

- The collection of servers, clients, connections between them, and any additional hardware and software that forms the global internetwork over which various types of data, including that which constitutes Web pages, travel.

link pseudo-class

- The pseudo-class type for attaching style to links based on the link's state with respect to user interaction with the links. Compare with dynamic pseudo-class.
- States include: 1.) :link, for attaching style to unvisited links; and 2.) :visited, for attaching style to visited links.

markup

- A.k.a., tags.
- An HTML/XHTML construct through which two types of structural information are attached to content; 1.) a structure type (e.g., heading, paragraph, etc.), and 2.) the demarcation of the starting and ending points for the application of the structure type.
- Typically consists of a less than sign "<", an element, optional attributes, and a greater than sign ">" to form an opening tag that demarcates the starting point for the application of the element's structure type and optional attributes to content, and a less than sign "<", a forward slash "/", an element, and a greater than sign ">" to form a closing tag that demarcates the ending point for the application of the element's structure type and optional attributes to content.
- Syntax: 1.) <element>content</element> [without an attribute], and 2.) <element attribute="value">content</element> [with an attribute].

nonstandard HTML extension

- Elements and attributes, typically created by Web Browser vendors, that are not defined by a W3C HTML/XHTML Recommendation.

obsoleted

- Elements and attributes that are no longer defined by a W3C HTML/XHTML Recommendation. Compare with deprecated.
- Elements and attributes that user agents might support, but are no longer encouraged to support.

parent

- An element in the same branch as, and one level above, another element in the HTML/XHTML document tree.
- An element that contains (i.e., encloses) one or more elements. Compare with child.

- The ancestor that is only one level above another element in the HTML/XHTML document tree.

property

- One of the two parts of a declaration. Compare with value.
- A characteristic of content defined by CSS.
- Examples: font-family, color, and font-size.

pseudo-class

- A selector. Compare with className and element.
- A mechanism defined by the W3C CSS Recommendations, reminiscent of the attribute, class, but HTML/XHTML independent (hence, "pseudo"-class) for attaching style to content based on the content's state with respect to user interaction with the content.
- Divided into two types; 1.) link pseudo-classes, and 2.) dynamic pseudo-classes.

Recommendation

- Short for W3C Recommendation.
- *"...a specification or set of guidelines that, after extensive consensus building, has received the endorsement of W3C Members and the Director. W3C recommends the wide deployment of its Recommendations. Note: W3C Recommendations are similar to the standards published by other organizations." -W3C.*
- A document published by the W3C that defines a Web language and its proper usage.

render

- To display the result of the application of HTML/XHTML structure and CSS style upon content.

rendering engine

- A.k.a., layout engine.
- The component of a user agent that renders.

selector

- One of the two parts of a style rule. Compare with declaration block.
- The element(s), and/or className(s), and/or pseudo-class(es) to which a style rule applies.
- The bridge (i.e., common denominator or link) between a style rule and an HTML/XHTML document via which style is attached to content.

sibling

- One of two or more elements one level below the same element in the HTML/XHTML document tree.
- One of two or more elements contained within (i.e., enclosed by) the same element.
- One of two or more children of the same parent.

source code

- The text, itself, that constitutes a Web document or Web page.

standard

- A W3C Recommendation or a document of a similar nature published by a different organization that defines a Web language and its proper usage.
- A W3C Recommendation or a document of a similar nature published by a different organization that is widely implemented by user agents, authors, and users.

structure

- The coarse layout/presentation of content as provided by HTML/XHTML. Compare with style.
- To coarsely layout/present content via HTML/XHTML.
- Includes headings, paragraphs, ordered lists, tables, forms, etc.
- A generic term referring to HTML/XHTML and/or elements.

style

- The refined layout/presentation of content as provided by CSS. Compare with structure.
- To refine the layout/presentation of content via CSS.
- Includes fonts, colors, borders, margins, padding, etc.
- Consists of a property and value pair, each of which constitutes a declaration.
- A generic term referring to CSS, and/or one or more style sheets, and/or one or more style rules, and/or one or more selectors, and/or one or more declaration blocks, and/or one or more declarations, and/or one or more properties, and/or one or more values.

style conflict

- Two or more styles attached to content in the same Web page having the same selector, the same property, but different values.
- Can occur: 1.) between the user agent default styles, and/or author styles list, and/or user styles list; and 2.) within the author styles list and/or user styles list.

style, HTML

- The attribute for attaching an inline style to content. Compare with class.
- An attribute added to the W3C HTML 4.0 Recommendation as part of its support for the W3C CSS1 Recommendation.

style rule

- A.k.a., rule.
- The basic CSS unit attached to content for the purpose of styling content.
- Consists of a selector and a declaration block.
- Syntax: selector { declaration block }.

style sheet

- A collection of style rules.
- Includes three types; 1.) embedded style sheets, 2.) external style sheets, and 3.) imported style sheets.

tag

- See markup.

URI

- Uniform Resource Identifier
- A more technically correct term for URL (Uniform Resource Locator).
- A string that identifies a resource (i.e., Web site, Web page, image, downloadable file, service, etc.) accessible over the Internet.
- Example: <http://www.stevestechresource.com/str/instructional/css.html>.

user

- A.k.a., reader.
- A person employing a user agent to interact with Web pages/sites. Compare with author and user agent.
- One of the three sources of style. See user styles.

user agent

- A.k.a., UA, Web Browser.
- An application (i.e., agent) employed by users to interact with Web pages/sites.
- An application that renders Web pages.
- One of the three sources of style. See user agent default styles.
- Examples: Internet Explorer, Firefox, Safari, and Opera.

user agent default styles

- Styles built into a user agent. Compare with author styles and user styles.
- Define and provide the structure types associated with HTML/XHTML elements.
- Are merged with author styles, if present, and user styles, if present, in styling content.

- In the absence of author styles and user styles, the only (i.e., default) styles applied to content.
- Typically partially editable via the user agent's Options/Preferences.

user styles

- Styles written by a user. Compare with author styles and user agent default styles.
- Allow users, particularly those with accessibility requirements, to style content.
- Require a user agent whose Options/Preferences allow the user to link their own external style sheet to content.
- Includes external style sheets and imported style sheets.

user styles list

- The list of user style rules associated with a Web page, replacing any @import statements with the external style sheets to be imported, themselves. Compare with author styles list.
- Created by the Cascade.

valid

- That which conforms with a Web language's proper usage as defined by a W3C Recommendation.

value

- One of the two parts of a declaration. Compare with property.
- Adds specificity to a property, thereby defining a style. Compare with value, HTML.
- Examples: arial, #000000, 12px.

value, HTML

- One of the two parts of an attribute.
- Adds specificity to an attribute, thereby defining an attribute. Compare with value.
- Examples: arial, #000000, 12px.

W3C

- The World Wide Web Consortium.
- *"An international consortium where Member organizations, a full-time staff, and the public work together to develop Web standards."* - W3C.
- Mission: *"To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web."* -W3C.

Web

- See WWW.

Web Browser

- See user agent.

Web Browser window

- The part of a user agent where rendered Web pages are displayed.

Web document

- A text file written in a one or more Web languages having a Web language's associated file extension.
- Examples: CSS documents (.css) and HTML/XHTML documents (.htm or .html).

Web language

- A language for creating Web documents.
- Examples: CSS, HTML, XHTML, JavaScript, Perl, PHP, Java.

Web page

- That which is displayed in the Web Browser window when a user agent is directed to a Web page/site.
- Consists of one or more Web documents. For example, a Web page might consist of multiple HTML/XHTML documents and multiple CSS documents.

Web page/site

- Web page and/or Web site.

Web site

- A location (i.e., site) on the WWW consisting of one or more Web pages.
- Example: www.stevestechresource.com.

WWW

- World Wide Web.
- A.k.a., Web.
- The collection of Web pages and Web sites accessible over the Internet.

XHTML

- Extensible HyperText Markup Language.
 - A reformulation of HTML as an XML application.
-

3. CSS Basics

A style rule is the basic CSS unit attached to content for the purpose of styling content. A style rule consists of a selector and a declaration block. Syntax:

```
selector { declaration block }
```

A selector is the element(s), and/or className(s), and/or pseudo-class(es) to which a style rule applies. A selector is the bridge (i.e., common denominator or link) between a style rule and an HTML/XHTML document via which style is attached to content.

A declaration block assigns one or more styles to a selector. A declaration block consists of one or more declarations, each declaration representing a style. Syntax:

```
selector { declarationOne; declarationTwo; declarationThree } [a declaration block with three declarations]
```

A declaration is a style. A declaration consists of a property and value pair. Syntax:

```
selector { propertyOne: valueOne; propertyTwo: valueTwo; propertyThree: valueThree } [a style rule with three declarations (i.e., styles)]
```

A property is a characteristic of content defined by CSS. A value adds specificity to a property, thereby defining a style.

A couple style rule examples:

```
h1 { font-style: italic; text-decoration: underline }  
.red { color: #ff0000 }  
:link { color: #000000 }
```

The selector is often used to refer to a style rule by name. For example:

```
h1 { font-style: italic; text-decoration: underline } [the heading one or h1 style rule]  
.red { color: #ff0000 } [the red, dot red, or .red style rule]  
:link { color: #000000 } [the link or :link style rule]
```

3.1. CSS Comments

A CSS comment is text in a style sheet that is not interpreted by, and, therefore, is ignored by, a user agent. Comments typically serve as a note/reminder (i.e., comment) about the surrounding, interpreted code. CSS comments are preceded by `"/*"` and followed by `"*/"` characters, no quotes. CSS comments are also single line, meaning, they can wrap, but cannot span multiple lines.

Syntax:

```
selector { declaration block } /* CSS comment at the end of a line. */
/* CSS comment at the start of a line. */
```

3.2. Grouping Selectors And Grouping Declarations

Grouping selectors and grouping declarations allows multiple style rules to be combined (i.e., grouped) into a single style rule. This reduces style sheet size.

Selectors with the same declaration block can be grouped into a comma separated selector. For example:

```
p { color: #0000ff }
.blue { color: #0000ff }
:hover { color: #0000ff }
```

Is equivalent to:

```
p, .blue, :hover { color: #0000ff }
```

Conversely, declarations with the same selector can be grouped into a semicolon separated declaration block. For example:

```
h2 { font-style: italic }
h2 { text-decoration: underline }
```

Is equivalent to:

```
h2 { font-style: italic; text-decoration: underline }
```

3.3. Valid Properties, Values, And The Handling Of Invalid Styles

The W3C CSS Recommendations define the valid properties for each element and group of related elements. For example, for the block-level elements, h1 and p, the properties, background-color and color, are valid, but the properties, vertical-align and empty-cells, are invalid.

The W3C CSS Recommendations also define the valid values for each property and group of related properties. For example, for the box model properties, margin-top and padding-top, the values, 7px and 0.5em, are valid, but the values, arial and italic, are invalid.

The W3C CSS Recommendations define a valid style as a declaration with a valid property, value, and syntax, and an invalid style as a declaration with an invalid property, value, or syntax. Moreover, conforming user agents are to respect valid styles and are to ignore invalid styles. In other words, validity occurs at the level of a style, not at the level of a style rule.

Consider, for example, the following style rules:

```
h1 { vertical-align: left; background-color: #000000; color: #00ff00 } [the first declaration's property is invalid]
h2 { font-family: arial; background-color: justify; color: #00ff00 } [the second declaration's value is invalid]
h3 { font-family: arial; background-color: #000000; color="#00ff00" } [the third declaration's syntax is invalid]
h4 { vertical-align: left; background-color: justify; color="#00ff00" } [all three declarations are invalid]
```

User agents are to interpret the style rules as:

```
h1 { background-color: #000000; color: #00ff00 }
h2 { font-family: arial; color: #00ff00 }
h3 { font-family: arial; background-color: #000000 }
h4 { }
```

Note: Some user agents are designed to treat invalid styles with common errors as if they were valid styles.

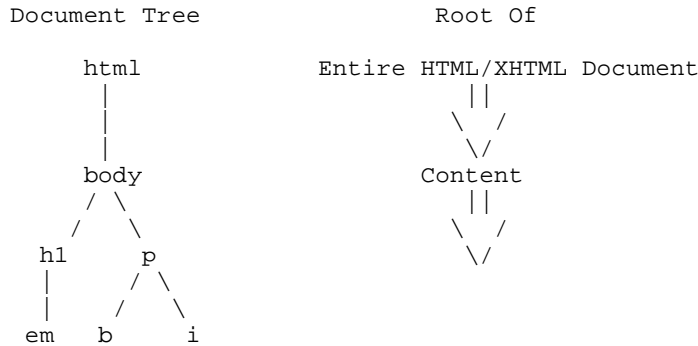
3.4. The HTML/XHTML Document Tree And The Inheritance Of Style

The HTML/XHTML document tree is a diagram of the elements that constitute an HTML/XHTML document. The diagram is similar in shape to a tree in that it has a root and branches. By convention, though, the root is placed at the top of the diagram.

Suppose, for example, that an HTML/XHTML document has the following source code:

```
<html>
<body>
<h1>Heading 1 With Some Text <em>Emphasized</em></h1>
<p>Paragraph with some text <b>bold</b> and <i>italic</i>.</p>
</body>
</html>
```

Its HTML/XHTML document tree is:



The element, `html`, is the root (i.e., origin) of the HTML/XHTML document, itself. The element, `body`, is the root of content.

An HTML/XHTML document tree depicts and is frequently used to define the terms, parent, child, sibling, ancestor, descendant, and inheritance.

A parent is an element in the same branch as, and one level above, another element in the HTML/XHTML document tree. For example, in the HTML/XHTML document tree above: 1.) the element, `html`, is the parent of the element, `body`; 2.) the element, `body`, is the parent of the elements, `h1` and `p`; 3.) the element, `h1`, is the parent of the element, `em`; and 4.) the element, `p`, is the parent of the elements, `b` and `i`. But, for example: 1.) the element, `h1`, is not the parent of the elements, `b` and `i`, because the element, `h1`, and the elements, `b` and `i`, are not in the same branch of the HTML/XHTML document tree; and 2.) the element, `body`, is not the parent of the elements, `em`, `b`, and `i`, because the element, `body`, is more than one level above the elements, `em`, `b`, and `i`, in the HTML/XHTML document tree.

Conversely, a child is an element in the same branch as, and one level below, another element in the HTML/XHTML document tree. For example, in the HTML/XHTML document tree above: 1.) the element, `body`, is the child of the element, `html`; 2.) the elements, `h1` and `p`, are the children of the element, `body`; 3.) the element, `em`, is the child of the element, `h1`; and 4.) the elements, `b` and `i`, are the children of the element, `p`. But, for example: 1.) the elements, `b` and `i`, are not the children of the element, `h1`, because the elements, `b` and `i`, and the element, `h1`, are not in the same branch of the HTML/XHTML document tree; and 2.) the elements, `em`, `b`, and `i`, are not the children of the element, `body`, because the elements, `em`, `b`, and `i`, are more than one level below the element, `body`, in the HTML/XHTML document tree.

Siblings are two or more elements one level below the same element in the HTML/XHTML document tree. For example, in the HTML/XHTML document tree above: 1.) the elements, `h1` and `p`, are siblings of the element, `body`; and 2.) the elements, `b` and `i`, are siblings of the element, `p`. But, for example, the elements, `em` and `b`, are not siblings because the elements, `em` and `b`, are more than one level below the same element in the HTML/XHTML document tree.

According to the definitions for parent, child, and sibling; 1.) a parent can have one or more children, 2.) a child has only one parent, and 3.) siblings are the children of the same parent. In other words, looking at the source code; 1.) a parent contains (i.e., encloses) one or more children, 2.) a child is contained within a single parent, and 3.) siblings are two or more children contained within the same parent.

An ancestor is an element in the same branch as, and one or more levels above, another element in the HTML/XHTML document tree. For example, in the HTML/XHTML document tree above: 1.) the element, `html`, is the ancestor of the elements, `body`, `h1`, `p`, `em`, `b`, and `i`; 2.) the element, `body`, is the ancestor of the elements, `h1`, `p`, `em`, `b`, and `i`; 3.) the element, `h1`, is the ancestor of the element, `em`; and 4.) the element, `p`, is the ancestor of the elements, `b` and `i`. But, for example, the element, `h1`, is not the ancestor of the elements, `b` and `i`, because the element, `h1`, and the elements, `b` and `i`, are not in the same branch of the HTML/XHTML document tree.

Conversely, a descendant is an element in the same branch as, and one or more levels below, another element in the HTML/XHTML document tree. For example, in the document tree above: 1.) the element, `body`, is the descendant of the element,

html; 2.) the elements, h1 and p, are the descendants of the elements, body and html; 3.) the element, em, is the descendant of the elements, h1, body, and html; and 4.) the elements, b and i, are the descendants of the elements, p, body, and html. But, for example, the elements, b and i, are not the descendants of the element, h1, because the elements, b and i, and the element, h1, are not in the same branch of the HTML/XHTML document tree.

According to the definitions for ancestor and descendant: 1.) ancestors include an element's parent, its parent's parent, its parent's parent's parent, etc.; and 2.) descendants includes an element's children, its children's children, and its children's children's children, etc.

One of the most important aspects of CSS is inheritance. Inheritance is the transferring of style from ancestors to descendants and the receiving of style by descendants from ancestors.

The W3C CSS Recommendations define which properties are inheritable and which are not inheritable. In other words, inheritance occurs at the level of a style, not at the level of a style rule. For the most part, whether or not a style is inheritable follows simple logic. For example, it makes sense that styles with the font properties, font-family, font-variant, font-style, font-weight, and font-size, are inheritable, but that styles with the box model properties, margin, padding, and border, are not inheritable. An inheritable style is inherited only if its property is valid for the descendant(s).

Consider, for example, the following code:

```
<p>Paragraph with some text <b>bold</b> and <i>italic</i>.</p>
```

Suppose, for example, that you want the entire paragraph, including the bold and italic text, styled with color red. One way to accomplish this is to assign the style, color red, to all three elements, p, b, and i. An easier way to accomplish this is to take advantage of inheritance. Notice in this example that the element, p, is the parent of the elements, b and i. Moreover, the property, color, is inheritable and is valid for the elements, b and i. Therefore, taking advantage of inheritance, the entire paragraph, including the bold and italic text, can be styled with color red simply by assigning the style, color red, to the parent element, p.

3.5. Important Styles

As described in The Cascade's Rules For Resolving Style Conflicts (below), by default, when style conflicts occur, author styles override user styles. To allow user styles to override author styles, the W3C CSS Recommendations define !important styles. Author styles and user styles can be marked !important. !important styles override normal (i.e., non-!important) styles, and !important user styles override !important author styles.

Note:

- That author styles can override user styles and, conversely, user styles can override author styles, defines author/user balance.
- In the W3C CSS1 Recommendation, !important author styles override !important user styles. In the W3C CSS2 Recommendation, !important user styles override !important author styles. Since the W3C CSS2 Recommendation supersedes the CSS1 Recommendation, current user agents should support !important user styles overriding !important author styles.

The marking of !important occurs at the level of a style, not at the level of a style rule. To mark a style !important, add "!important" or "! important," no quotes, to the end of the style's declaration. For example, in the following style rule, the styles, font family arial, padding 0 pixels, and border 2 pixels red solid are marked !important:

```
body { font-family: arial !important; font-size: 15px; padding: 0px ! important; background-color: #eeeeee; border: 2px #ff0000 solid !important }
```

A style with a shorthand property marked !important marks all of its subproperties !important.

3.6. Style Rule Syntax And Conventions

Note: Rather than remembering the rules for case-sensitivity described below, since valid XHTML requires elements and attributes in lowercase letters (a-z), it is recommended that you simply use lowercase letters throughout your markup and style sheets except for when you want to use multiple concatenated words for the value assigned to the attribute, class, which corresponds to the selector, className, in which case it is recommended that you use lowercase letters for the first word and an uppercase letter (A-Z) for the first letter of each subsequent word (i.e., class="wordWordWord" and .wordWordWord). Not only does this convention conform with valid XHTML, but valid CSS as well.

- The order of style rules in a style sheet does not matter except for the order of style rules with pseudo-classes as selectors.

Note: Due to [The Cascade's Rules For Resolving Style Conflicts \(below\)](#), specifically, if two [styles](#) have the same weight and origin, the latter specified wins, it is strongly recommended that [style rules](#) with [pseudo-classes](#) as [selectors](#) are listed in your [style sheets](#) in the following order: `:link` before `:visited` before `:hover` before `:active` before `:focus`. If they are listed in a different order, a style assigned to a pseudo-class listed earlier in your style sheets might not be applied to [content](#) as expected because it is being overridden by a style assigned to a pseudo-class listed later in your style sheets.

- [Elements](#) and pseudo-classes are not case-sensitive (i.e., `body` and `BODY` are equivalent and `:link` and `:LINK` are equivalent).
- The [value](#) assigned to the [attribute](#), [class](#), which corresponds to the selector, [className](#):
 - Are case-sensitive (i.e., `class="red"` and `.RED` and not equivalent).
 - Can contain letters (a-z A-Z), digits (0-9), and the characters underscore (`_`) and hyphen (`-`). Spaces and line breaks are not allowed.
 - The first character must be a letter (a-z A-Z). The first character cannot be a digit (0-9) or the characters underscore (`_`) or hyphen (`-`).
 - Use a single word or multiple concatenated words that preferably describe the [content](#), not the [style](#). For example, if keywords are to be styled with background color powderblue, use `class="keyword"` and `.keyword`, not `class="powderblue"` and `.powderblue`. The reason for describing the content instead of the style is that if you decide to change the style for keywords to, for example, background color green, the value assigned to the attribute, class, which corresponds to the selector, [className](#), would no longer reflect the style, which could be confusing.
- [ClassNames](#) are preceded by a period (`.`).
- [Pseudo-classes](#) are preceded by a colon (`:`).
- [Properties](#) and values are not case-sensitive (i.e., `font-family: arial` and `FONT-FAMILY: ARIAL` are equivalent).
- Spaces and line breaks before and after [declarations](#), properties, values, and `!important` are optional (i.e., `{font-family:arial!important;font-size:2em}` and `{ font-family : arial !important ; font-size : 2em }` are equivalent).
- [Declarations](#) are separated by a semicolon (`;`).
- The order of declarations does not matter.
- A semicolon (`;`) after a [declaration block's](#) only or last declaration is optional (i.e., `{ font-family: arial }` and `{ font-family: arial; }` are equivalent).
- Properties and values are separated by a colon (`:`).
- A space between `!` and `important` is optional (i.e., `!important` and `! important` are equivalent).
- The most common style rule convention:

```
selector {
  propertyOne: valueOne;
  propertyTwo: valueTwo;
  propertyThree: valueThree;
}
```

Note: I dislike the [style rule](#) convention above because of the vertical scrolling induced by all the line breaks.

4. Creating Style Rules And Attaching Styles To Content

A [style rule](#) is the basic [CSS](#) unit attached to [content](#) for the purpose of [styling](#) content. A style rule consists of a [selector](#) and a [declaration block](#). Syntax:

```
selector { declaration block }
```

A selector is the [element\(s\)](#), and/or [className\(s\)](#), and/or [pseudo-class\(es\)](#) to which a style rule applies. A selector is the bridge (i.e., common denominator or link) between a style rule and an [HTML/XHTML document](#) via which style is attached to content.

A declaration block assigns one or more styles to a selector. A declaration block consists of one or more [declarations](#), each declaration representing a style. Syntax:

```
selector { declarationOne; declarationTwo; declarationThree } [a declaration block with three declarations]
```

A declaration is a style. A declaration consists of a [property](#) and [value](#) pair. Syntax:

```
selector { propertyOne: valueOne; propertyTwo: valueTwo; propertyThree: valueThree } [a style rule with three declarations]
```

(i.e., styles)]

A property is a characteristic of content defined by CSS. A value adds specificity to a property, thereby defining a style.

4.1. Elements As Selectors

An element is an HTML/XHTML structure type attached to content through the use of markup. Elements, including the element, body, and all of its descendants, can serve as selectors.

4.1.1. The Element, Body, As Selector And Defining A Default Style For A Web Page/Site

The element, body, is the root (i.e., origin) and ancestor of all content. Due to inheritance, the styles attached to the element, body, that are inheritable and valid for its descendants, are inherited by its descendants, and, therefore, define a default style for a Web page/site.

To maximize the benefit of inheritance, the body style rule should be assigned the styles desired for the majority of a Web page's/site's content. With the styles for the majority of content accounted for by the body style rule, the only remaining style rules needed are for the minority of content whose styles are to deviate from the Web page's/site's default style.

A style rule with the element, body, as selector. Syntax:

```
body { declaration block }
```

Suppose, for example, that you want the majority of a Web page's/site's content styled with font family arial, background color lightgrey, color black, and font size 15 pixels. The following style rule provides this:

```
body { font-family: arial; background-color: #d3d3d3; color: #000000; font-size: 15px }
```

To attach the styles to content, use the element, body, in markup. Syntax:

```
<body>content</body>
```

The styles attached to content:

```
<body>
body styled via the body style rule.
<h1>h1 styled via the body style rule.</h1>
<p>p styled via the body style rule.</p>
</body>
```

The styles applied to content: 4.1.1. The Element, Body, As Selector And Defining A Default Style For A Web Page/Site (stevestechresource.com).

4.1.2. Elements Other Than The Element, Body, As Selectors

Elements other than the element, body, not only can inherit styles from the body style rule, but can serve as selectors themselves.

A style rule with an element other than the element, body, as selector. Syntax:

```
element { declaration block }
```

Suppose, for example, that in addition to wanting the majority of a Web page's/site's content styled with font family arial, background color lightgrey, color black, and font size 15 pixels, you also want h1 headings styled with font style italic and background color white, and paragraphs styled with margin left 20 pixels. The following style rules provide this:

```
body { font-family: arial; background-color: #d3d3d3; color: #000000; font-size: 15px }
h1 { font-style: italic; background-color: #ffffff }
p { margin-left: 20px }
```

To attach the styles to content, use the elements in markup. Syntax:

```
<element>content</element>
```

The styles attached to content:

```

<body>
body styled via the body style rule.
<h1>h1 styled via the body and h1 style rules.</h1>
<p>p styled via the body and p style rules.</p>
</body>

```

The styles applied to content: [4.1.2. Elements Other Than The Element, Body, As Selectors \(stevestechresource.com\)](#).

4.2. ClassNames As Selectors

Style rules with elements as selectors bind elements and styles to one another. As a result, using elements as selectors is rather restrictive for styling content in that an element's style is fixed. To provide flexibility for styling content, the [W3C HTML 4.0 Recommendation](#) defines the [attribute](#), [class](#), and the [W3C CSS1 Recommendation](#) defines the [value](#) assigned to it, [className](#), as a selector.

Note: The [attribute](#), [class](#), was added to the [W3C HTML 4.0 Recommendation](#) as part of its support for the [W3C CSS1 Recommendation](#). The [attribute](#), [class](#), behaves different than other [HTML/XHTML](#) attributes in that the [value](#) assigned to the [attribute](#), [class](#), declares a variable whose name is the value assigned to the [attribute](#), [class](#), and whose value is a [declaration block](#). That the value assigned to the [attribute](#), [class](#), behaves like a variable name is why [className](#), rather than, for example, [classValue](#), is used in this [Web page](#) to represent the value assigned to the [attribute](#), [class](#).

ClassName is an [author](#)-defined variable that functions as a selector. Style rules with classNames as selectors separate elements and styles from one another by placing an intermediary, the selector, [className](#), between them. As a result, using classNames as selectors provides the following flexibility for styling content; 1.) each instance of an element can have a different style attached to it, and, conversely, 2.) a style can be attached to different elements.

A style rule with [className](#) as selector. Syntax:

```
.className { declaration block }
```

Suppose, for example, that in addition to wanting the majority of a [Web page's/site's](#) content styled with font family arial, background color lightgrey, color black, and font size 15 pixels, h1 headings styled with font style italic and background color white, and paragraphs styled with margin left 20 pixels, you also want some of the h1 headings and paragraphs styled with color red or color blue. The following style rules provide this:

```

body { font-family: arial; background-color: #d3d3d3; color: #000000; font-size: 15px }
h1 { font-style: italic; background-color: #ffffff }
p { margin-left: 20px }
.red { color: #ff0000 }
.blue { color: #0000ff }

```

To attach the styles to content, use the elements, h1 and p, with the [attribute](#), [class](#), assigned the value, [className](#), in [markup](#). Syntax:

```
<element class="className">content</element>
```

An element with the [attribute](#), [class](#), assigned the value, [className](#), is known as a [classed element](#). All of the [descendants](#) of the element, [body](#), can be classed.

The styles attached to content:

```

<body>
body styled via the body style rule.
<h1>h1 styled via the body and h1 style rules.</h1>
<p>p styled via the body and p style rules.</p>
<h1 class="red">h1 styled via the body, h1, and .red style rules.</h1>
<p class="red">p styled via the body, p, and .red style rules.</p>
<h1 class="blue">h1 styled via the body, h1, and .blue style rules.</h1>
<p class="blue">p styled via the body, p, and .blue style rules.</p>
</body>

```

The styles applied to content: [4.2. ClassNames As Selectors \(stevestechresource.com\)](#).

4.2.1. Classing The Generic Inline Element, Span, And The Generic Block-Level Element, Div

The [W3C HTML/XHTML Recommendations](#) define two generic elements: 1.) a generic [inline element](#), span, and 2.) a generic [block-level element](#), div. Generic, in this context, means stripped of all [style](#) except that required to define the elements, span and div, as an inline element and block-level element, respectively. In other words, the element, span, is a minimally [structured](#) inline element, and the element, div, is a minimally structured block-level element.

Being blank canvases, so to speak, the generic elements, span and div, are ideally suited for styling [content](#) when removing the structural information from a structured (i.e., nongeneric) element is counterproductive or cannot be accomplished to satisfaction.

Suppose, for example, that you want keywords styled with background color powderblue. The following [style rule](#) provides this:

```
.keyword { background-color: #b0e0e6 }
```

[Classing](#) the generic inline element, span, is ideally suited for styling keywords. To class the generic inline element, span, use the element, span, with the [attribute](#), [class](#), assigned the [value](#), [className](#), in [markup](#). The style attached to content:

```
<p>The keyword in this sentence is <span class="keyword">abc</span> styled via the .keyword style rule.</p>
```

The style applied to content: [4.2.1. Classing The Generic Inline Element, Span \(stevestechresource.com\)](#).

Suppose, for example, that you want to create a tableless three column [Web page](#) layout. The following style rules provide this (background-color and height added for aesthetics):

```
.sidebarLeft { float: left; width: 20%; background-color: #ffcccc; height: 500px }
.main { float: left; width: 60%; background-color: #ccffcc; height: 500px }
.sidebarRight { float: left; width: 20%; background-color: #ccccff; height: 500px }
```

Classing the generic block-level element, div, is ideally suited for creating a tableless multicolumn Web page layout. To class the generic block-level element, div, use the element, div, with the [attribute](#), [class](#), assigned the [value](#), [className](#), in [markup](#). The styles attached to content:

```
<div class="sidebarLeft">Left sidebar styled via the .sidebarLeft style rule.</div>
<div class="main">Main section styled via the .main style rule.</div>
<div class="sidebarRight">Right sidebar styled via the .sidebarRight style rule.</div>
```

The styles applied to content: [4.2.1. Classing The Generic Block-Level Element, Div \(stevestechresource.com\)](#).

Important Note: Web design using [structured](#) (i.e., nongeneric) elements is known as designing by structure. Web design using [classed](#) generic elements, span and div, is known as designing by [style](#). Despite the flexibility and control that designing by style provides, it is strongly recommended that [authors](#) design by structure; 1.) it ensures that your [Web pages](#) are structurally presentable in [user agents](#) that do not support style or have support for style disabled, and 2.) [user styles](#) are practical only if the structured elements are present and can be used as [selectors](#).

4.3. Pseudo-Classes As Selectors

The [W3C CSS Recommendations](#) define a mechanism for attaching [style](#) to content based on the content's state with respect to [user](#) interaction with the content. This mechanism, reminiscent of the [attribute](#), [class](#), but [HTML/XHTML](#) independent, uses [selectors](#) known as [pseudo-classes](#). The [W3C CSS2 Recommendation](#) divides pseudo-classes into two types; 1.) [link pseudo-classes](#), and 2.) [dynamic pseudo-classes](#).

The [W3C CSS2 Recommendation](#) defines two link pseudo-classes. The link pseudo-classes attach style to links based on the link's state with respect to user interaction with the links as follows:

Link Pseudo-Class	Link State
:link	Unvisited.
:visited	Visited.

The [W3C CSS2 Recommendation](#) defines three dynamic pseudo-classes. The dynamic pseudo-classes attach style to content (which, by default, includes links) based on the content's state with respect to user interaction with the content as follows:

Dynamic Pseudo-Class	Content State
----------------------	---------------

:hover	With the mouse cursor placed over it.
:active	Between the mouse down click and the mouse up click.
:focus	Ready to accept input.

Style rules with pseudo-classes as selectors. Syntax:

```
:link { declaration block }
:visited { declaration block }
:hover { declaration block }
:active { declaration block }
:focus { declaration block }
```

Which, because user agents attach the pseudo-classes to links by default, are equivalent to:

```
a:link { declaration block }
a:visited { declaration block }
a:hover { declaration block }
a:active { declaration block }
a:focus { declaration block }
```

To assign a style to all links regardless their state, either assign the style to each pseudo-class, or assign the style to the element, a.

Suppose, for example, that you want all links styled with font style italic, unvisited links styled with text decoration underline, visited links styled with color silver and text decoration underline, hover links styled with color silver and text decoration none, and active links styled with color lime and text decoration none. The following style rules provide this:

```
a { font-style: italic }
a:link { text-decoration: underline }
a:visited { color: #c0c0c0; text-decoration: underline }
a:hover { color: #c0c0c0; text-decoration: none }
a:active { color: #00ff00; text-decoration: none }
```

Note: Due to The Cascade's Rules For Resolving Style Conflicts (below), specifically, if two styles have the same weight and origin, the latter specified wins, it is strongly recommended that style rules with pseudo-classes as selectors are listed in your style sheets in the following order: :link before :visited before :hover before :active before :focus. If they are listed in a different order, a style assigned to a pseudo-class listed earlier in your style sheets might not be applied to content as expected because it is being overridden by a style assigned to a pseudo-class listed later in your style sheets.

To attach the styles to content, use the element, a, with the attribute, href, assigned the value, URI, in markup. Syntax:

```
<a href="URI">Link text.</a>
```

Note: The pseudo-classes are HTML/XHTML independent and, therefore, do not appear in markup.

The styles attached to content:

```
<a href="dummy_web_document1.html">Link 1 styled via the a, a:link, a:visited, a:hover, and a:active style rules.</a>
<a href="dummy_web_document2.html">Link 2 styled via the a, a:link, a:visited, a:hover, and a:active style rules.</a>
<a href="dummy_web_document3.html">Link 3 styled via the a, a:link, a:visited, a:hover, and a:active style rules.</a>
```

The styles applied to content: 4.3. Pseudo-Classes As Selectors Example #1 (stevestechresource.com).

As described in ClassNames As Selectors (above), all of the descendants of the element, body, can be classed. This includes links styled with the pseudo-classes.

Suppose, for example, that in addition to wanting all links styled with font style italic, unvisited links styled with text decoration underline, visited links styled with color silver and text decoration underline, hover links styled with color silver and text decoration none, and active links styled with color lime and text decoration none, you also want the unvisited links styled with color black, color red, or color blue. The following style rules provide this:

```
a { font-style: italic }
a:link { text-decoration: underline }
a:visited { color: #c0c0c0; text-decoration: underline }
a:hover { color: #c0c0c0; text-decoration: none }
a:active { color: #00ff00; text-decoration: none }
.black { color: #000000 }
.red { color: #ff0000 }
.blue { color: #0000ff }
```

To attach the styles to content, use the element, `a`, with the attribute, `href`, assigned the value, URI, and with the attribute, `class`, assigned the value, `className`, in markup. Syntax:

```
<a href="URI" class="className">Link text.</a>
```

The styles attached to content:

```
<a href="dummy_web_document4.html" class="black">Link 4 styled via the a, a:link, a:visited, a:hover, a:active, and .black style rules.</a>
<a href="dummy_web_document5.html" class="black">Link 5 styled via the a, a:link, a:visited, a:hover, a:active, and .black style rules.</a>
<a href="dummy_web_document6.html" class="black">Link 6 styled via the a, a:link, a:visited, a:hover, a:active, and .black style rules.</a>
<a href="dummy_web_document7.html" class="red">Link 7 styled via the a, a:link, a:visited, a:hover, a:active, and .red style rules.</a>
<a href="dummy_web_document8.html" class="red">Link 8 styled via the a, a:link, a:visited, a:hover, a:active, and .red style rules.</a>
<a href="dummy_web_document9.html" class="red">Link 9 styled via the a, a:link, a:visited, a:hover, a:active, and .red style rules.</a>
<a href="dummy_web_document10.html" class="blue">Link 10 styled via the a, a:link, a:visited, a:hover, a:active, and .blue style rules.</a>
<a href="dummy_web_document11.html" class="blue">Link 11 styled via the a, a:link, a:visited, a:hover, a:active, and .blue style rules.</a>
<a href="dummy_web_document12.html" class="blue">Link 12 styled via the a, a:link, a:visited, a:hover, a:active, and .blue style rules.</a>
```

The styles applied to content: [4.3. Pseudo-Classes As Selectors Example #2 \(stevestechresource.com\)](#).

4.4. Inline Styles

An inline style is a style, style rule and style sheet independent, attached to content via the attribute, style, assigned a value whose syntax mimics that of a declaration block, minus the braces.

An inline style. Syntax:

```
<element style="property: value">content</element>
```

Important Note: With inline styles, the styles are placed in markup, which is the least efficient way to style content. Therefore, it is strongly recommended that you do not use inline styles.

Suppose, for example, that you want a paragraph styled with text align center, font size 50 pixels, and margin top 100 pixels. The following inline style accomplishes this:

```
<p style="text-align: center; font-size: 50px; margin-top: 100px">p styled via an inline style.</p>
```

The styles applied to content: [4.4. Inline Styles \(stevestechresource.com\)](#).

5. Style Sheets

A style sheet is a collection of style rules. The W3C HTML and CSS Recommendations define three types of style sheets; 1.) embedded style sheets, 2.) external style sheets, and 3.) imported style sheets. All three types of style sheets can simultaneously style content.

5.1. Embedded Style Sheets

An embedded style sheet is a collection of style rules located inside (i.e., embedded within) an HTML/XHTML document. Specifically, an embedded style sheet is one or more style rules inside `<style type="text/css"></style>` tags placed in the `<head></head>` section of an HTML/XHTML document. Syntax:

```
<html>
<head>
<title></title>
<style type="text/css">
...list style rules here...
</style>
</head>
<body>
...content here...
</body>
</html>
```

The styles of an embedded style sheet are available to style, and are only available to style, the content of the HTML/XHTML document in which the embedded style sheet is located.

5.2. External Style Sheets

An external style sheet is collection of style rules located outside (i.e., external to) an HTML/XHTML document. Specifically, an external style sheet is one or more style rules in a text file having a `.css` file extension.

For the styles of an external style sheet to style content, the external style sheet must be associated with (i.e., linked to) an HTML/XHTML document. To link an external style sheet to an HTML/XHTML document, add the line, `<link rel="stylesheet" type="text/css" href="anyFileName.css" />`, where the value assigned to the attribute, `href`, is an absolute or relative path to an external style sheet, `anyFileName.css`, to the `<head></head>` section of an HTML/XHTML document as follows:

```
<html>
<head>
<title></title>
<link rel="stylesheet" type="text/css" href="anyFileName.css" />
</head>
<body>
...content here...
</body>
</html>
```

An external style sheet can be linked to multiple HTML/XHTML documents, including those of an entire Web site. To link an external style sheet to multiple HTML/XHTML documents, add the line, `<link rel="stylesheet" type="text/css" href="anyFileName.css" />`, where the value assigned to the attribute, `href`, is an absolute or relative path to an external style sheet, `anyFileName.css`, to the `<head></head>` section of each HTML/XHTML document.

Conversely, multiple external style sheets can be linked to an HTML/XHTML document. To link multiple external style sheets to an HTML/XHTML document, add the line, `<link rel="stylesheet" type="text/css" href="anyFileName.css" />`, where the value assigned to the attribute, `href`, is an absolute or relative path to an external style sheet, `anyFileName.css`, to the `<head></head>` section of an HTML/XHTML document for each external style sheet to be linked. In the following example, three external style sheets, `anyFileNameOne.css`, `anyFileNameTwo.css`, and `anyFileNameThree.css`, are linked to an HTML/XHTML document:

```
<html>
<head>
<title></title>
<link rel="stylesheet" type="text/css" href="anyFileNameOne.css" />
<link rel="stylesheet" type="text/css" href="anyFileNameTwo.css" />
<link rel="stylesheet" type="text/css" href="anyFileNameThree.css" />
</head>
<body>
...content here...
</body>
</html>
```

Note: The ability to link multiple external style sheets to an HTML/XHTML document provides modularity (i.e., authors, instead of needing to create a single large style sheet encompassing an entire Web page/site, can think of their Web pages/sites as consisting of sections (i.e., modules) each getting its own module-specific style sheet).

The styles of an external style sheet are available to style the content of each HTML/XHTML document to which the external style sheet is linked.

5.3. The Advantage Of External Style Sheets Over Embedded Style Sheets

Suppose, for example, that your Web site has 50 Web pages and you want the majority of the Web site's content styled with font family arial. A body style rule assigned the style, font family arial, provides this.

Implementing this style site wide with embedded style sheets involves creating an embedded style sheet containing the body style rule, and then pasting the embedded style sheet into the <head></head> section of each of the Web site's 50 Web pages. Implementing this style site wide with external style sheets involves creating an external style sheet, anyFileName.css, containing the body style rule, and then pasting the line, <link rel="stylesheet" type="text/css" href="anyFileName.css" />, where the value assigned to the attribute, href, is an absolute or relative path to the external style sheet, anyFileName.css, into the <head></head> section of each of the Web site's 50 Web pages. Hence, the amount of work required to implement the style site wide with external style sheets or embedded style sheets is roughly the same.

But what if you decide to change the majority of the Web site's content from font family arial to font family verdana? With embedded style sheets, because styles are embedded within each Web page, this change would involve editing each of the Web site's 50 Web pages. With external style sheets, because styles are centrally located in a single file, this change would involve editing just one file, the external style sheet, anyFileName.css. Hence, the advantage of external style sheets over embedded style sheets: editing styles site wide is much easier with external style sheets than embedded style sheets.

5.4. Imported Style Sheets

An imported style sheet is an external style sheet inserted (i.e., imported) into an embedded style sheet, external style sheet, or imported style sheet. Only external style sheets can be imported. The W3C CSS Recommendations define an @import statement that: 1.) indicates an absolute or relative path to the external style sheet, anyFileName.css, to be imported; and 2.) indicates where the imported style sheet is to be inserted, which is in the place of the @import statement, itself. Syntax:

```
@import "anyFileName.css";
```

Which is equivalent to:

```
@import url("anyFileName.css");
```

Multiple external style sheets can be imported into a style sheet. To import multiple external style sheets into a style sheet, create an @import statement for each external style sheet to be imported.

The @import statements function only if they are listed at the top of a style sheet. In other words, user agents are to respect @import statements listed at the top of a style sheet and are to ignore @import statements listed in a style sheet after a style rule. Consider, for example, the five @import statements in the following embedded style sheet:

```
<html>
<head>
<title></title>
<style type="text/css">
@import "anyFileNameOne.css"; [functions]
@import "anyFileNameTwo.css"; [functions]
body { font-family: arial; background-color: #ffffff; color: #000000; font-size: 15px }
h1 { text-decoration: underline }
@import "anyFileNameThree.css"; [ignored]
p { margin-left: 20px }
.red { color: #ff0000 }
@import "anyFileNameFour.css"; [ignored]
@import "anyFileNameFive.css"; [ignored]
.blue { color: #0000ff }
</style>
</head>
<body>
...content here...
```

```
</body>
</html>
```

The first and second @import statements above are listed at the top of the style sheet and, therefore, function. The third, fourth, and fifth @import statements above are listed after a style rule (the body style rule) and, therefore, are ignored.

An imported style sheet can have one or more external style sheets imported into it, which can have one or more external style sheets imported into it, which can have one or more external style sheets imported into it, etc.

6. User Agent Default Styles, Author Styles, And User Styles

The [W3C CSS Recommendations](#) define three sources of style; 1.) user agents, 2.) authors, and 3.) users. All three sources of style can simultaneously style content.

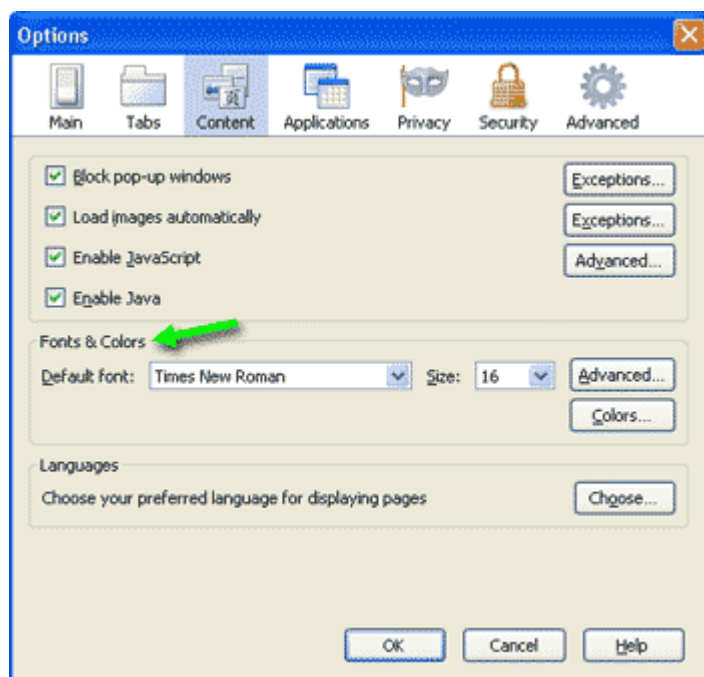
6.1. User Agent Default Styles

Few, including those who consider themselves [CSS](#) and/or [HTML/XHTML](#) experts, realize that user agents have a style sheet built into them whose styles: 1.) define and provide the structure types associated with [HTML/XHTML elements](#); 2.) are merged with author styles, if present, and user styles, if present, in styling content; and 3.) in the absence of author styles and user styles, are the only (i.e., default) styles applied to content; hence, the term, user agent default styles.

Of all the valid properties for a given element, user agent default styles use only those required to impart a basic structure type upon an element. In this respect, it can be said that [HTML/XHTML](#) is a Web language in which a minimum of styles, defined and provided by user agents, crudely layout (i.e., structure) content, and [CSS](#) is a Web language in which additional styles, defined and provided by authors and users, refine the layout of (i.e., style) content.

The [W3C CSS2 Recommendation](#) includes a Default style sheet for [HTML 4](#) that user agent vendors are encouraged to implement. User agent vendors, however, are free to create their own user agent default styles. For example, the user agent default styles of Opera 6-8 appear to include an ~8 pixel padding assigned to its body style rule, and the user agent default styles of Internet Explorer 5-7 appear to include an ~5 pixel padding top assigned to its list style types disc, circle, square, and none. User agents whose user agent default styles differ significantly from the norm, however, have a negative impact on [Cross-Browser compatibility](#) and are denounced by authors and users. Accordingly, the differences between user agent default styles have gradually decreased over time.

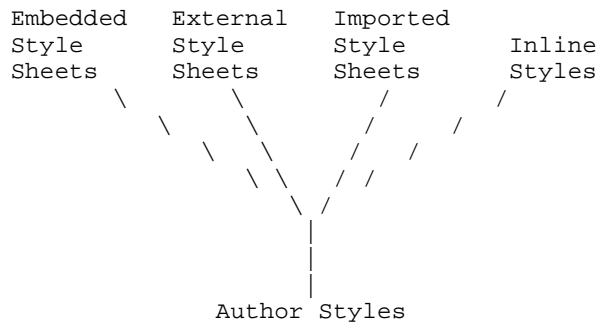
Besides acquiring user agent default styles from their built-in style sheet, most user agents acquire additional user agent default styles from their Options/Preferences. These additional user agent default styles, which users can change, typically include fonts and colors for the element, body, and the link pseudo-classes. For example, in Firefox 3.5, click Tools | Options | Content. The Fonts & Colors fieldset appears:



Note: In some user agents, the Options/Preferences related to style behave as if editing the user agent default styles. In other user agents, the Options/Preferences related to style behave as if instantiating user styles with the nondefault styles selected marked !important. Lastly, in some user agents, the Options/Preferences related to style include a setting that appears to toggle whether the Options/Preferences related to style behave as if editing the user agent default styles, or as if instantiating user styles with the nondefault styles selected marked !important. Further discussion of this topic is beyond the scope of this page.

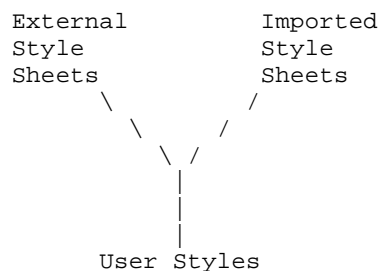
6.2. Author Styles

Author styles are styles written by an author. As illustrated below, author styles include embedded style sheets, external style sheets, imported style sheets, and inline styles:

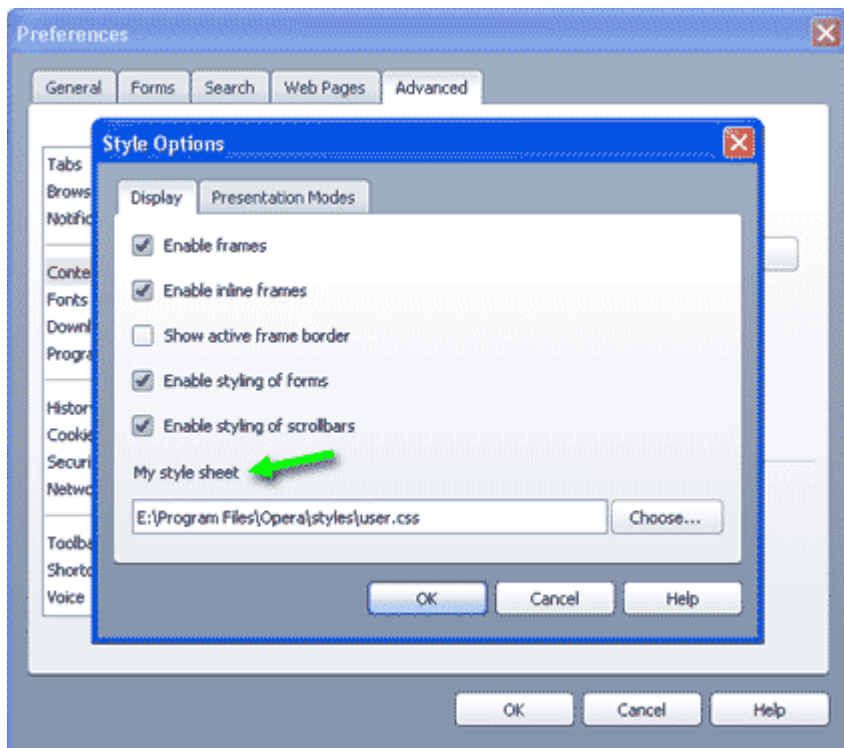


6.3. User Styles

User styles are styles written by a user. As illustrated below, user styles include external style sheets and imported style sheets:



User styles allow users, particularly those with accessibility requirements, to style content. User styles require a user agent whose Options/Preferences allow the user to link their own external style sheet to content. For example, in Opera 10.00, click Tools | Preferences | Advanced | Content | Style Options. The My style sheet field appears:



7. The Cascade

The most important but least understood aspect of CSS is the Cascade. The Cascade is a component of a user agent. In short, for each Web page, the Cascade: 1.) processes author styles, if present, into an author styles list, and user styles, if present, into a user styles list; 2.) merges the user agent default styles, author styles list, if present, and user style list, if present; 3.) resolves style conflicts; and 4.) creates a master style sheet whose styles the user agent's rendering engine applies to content in the Web Browser window.

The Cascade also: 1.) allows user agent default styles, author styles, and user styles to simultaneously style content; 2.) allows author styles and user styles to be partial styles (i.e., author styles and user styles need only assign the styles to add to, or change from, the user agents default styles); 3.) provides modularity (i.e., authors, instead of needing to create a single large style sheet encompassing an entire Web page/site, can think of their Web pages/sites as consisting of sections (i.e., modules) each getting its own module-specific style sheet); and 4.) provides author/user balance (i.e., a mechanism for author styles to override user styles and, conversely, for user styles to override author styles).

The W3C CSS Recommendations define the Cascade. User agent vendors, however, are free to create their own version of the Cascade. User agents whose version of the Cascade differs significantly from the norm, however, have a negative impact on Cross-Browser compatibility and are denounced by authors and users. Accordingly, the differences between user agent versions of the Cascade have gradually decreased over time.

Perhaps some of the misunderstandings about the Cascade comes from the W3C's own language. Consider, for example, the following two sentences describing one of the Cascade's rules for resolving style conflicts, where "reader" means user, and "UA's default values" and "the default style sheet" mean user agent default styles:

"Sort by origin: the author's style sheets override the reader's style sheet which override the UA's default values." - W3C CSS1 Recommendation

"The primary sort of the declarations is by weight and origin: for normal declarations, author style sheets override user style sheets which override the default style sheet." - W3C CSS2 Recommendation

The problem with the W3C language above is that the Cascade performs overriding at the level of a style, not at the level of a style sheet as the W3C language above suggests. In fact, if the Cascade performed overriding at the level of a style sheet, as the W3C language above suggests, then author styles and user style would cancel out the user agent default styles that define and provide the structure types associated with HTML/XHTML elements, and, as a result, author styles and user styles could not be partial styles for they would need to include the user agent default styles they just cancelled out.

Similarly, consider the following two sentences describing another one of the Cascade's rules for resolving style conflicts, where

"rules" means style rules:

"Sort by order specified: if two rules have the same weight, the latter specified wins." - W3C CSS1 Recommendation

"Finally, sort by order specified: if two rules have the same weight, origin and specificity, the latter specified wins." - W3C CSS2 Recommendation

The problem with the W3C language above is that the Cascade picks winners at the level of a style, not at the level of a style rule as the W3C language above suggests. In fact, if the Cascade picked winners at the level of a style rule, as the W3C language above suggests, then author styles and user styles would cancel out the user agent default styles that define and provide the structure types associated with HTML/XHTML elements, and, as a result, author styles and user styles could not be partial styles for they would need to include the user agent default styles they just cancelled out.

Besides the misunderstandings about the Cascade that might arise from the W3C's own language, many who consider themselves CSS experts spread misconceptions about the Cascade. The most common misconception about the Cascade is that the styles of embedded style sheets, being "closer to content," necessarily override the styles of external style sheets, which are "further from content."

The Cascade is not that difficult to understand. It is unfortunate, though, the poor language and misconceptions that have been associated with it. So, before starting, it is strongly recommended that you disabuse yourself of any preconceived notions about the Cascade so that a clear understanding can be built unencumbered from scratch.

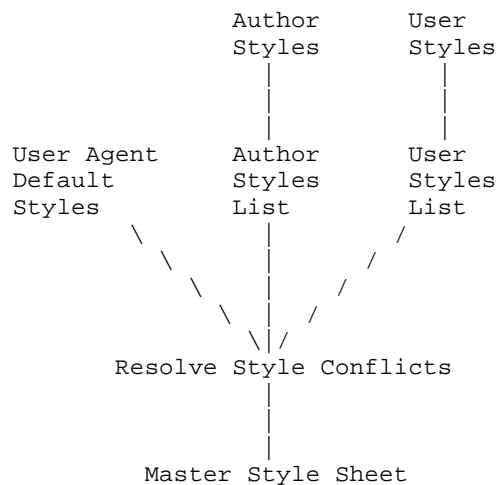
7.1. Overview Of The Cascade

The following diagrams provide an overview of the Cascade. For each Web page, the Cascade: 1.) processes author styles, if present, into an author styles list, and user styles, if present, into a user styles list; 2.) merges the user agent default styles, author styles list, if present, and user style list, if present; 3.) resolves style conflicts; and 4.) creates a master style sheet.

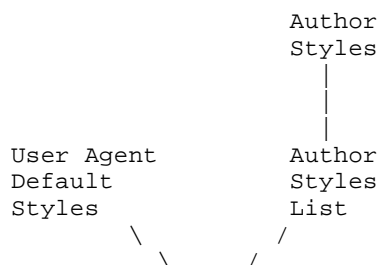
Note: For information on:

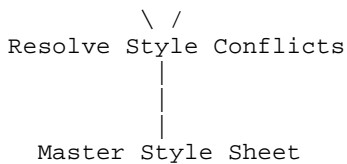
- user agent default styles, see User Agent Default Styles (above).
- author styles, see Author Styles (above).
- user styles, see User Styles (above).

7.1.1. The Cascade By Diagram: Author Styles Present. User Styles Present.

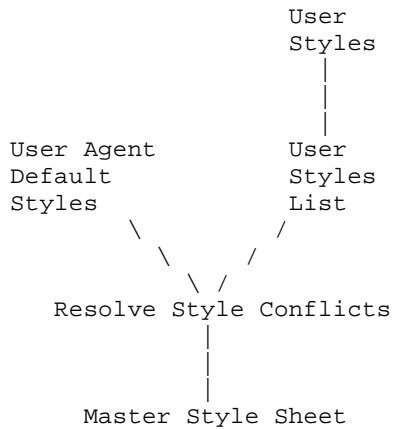


7.1.2. The Cascade By Diagram: Author Styles Present. User Styles Absent.

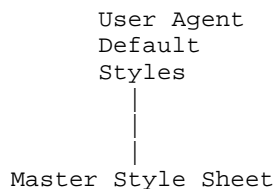




7.1.3. The Cascade By Diagram: Author Styles Absent. User Styles Present.



7.1.4. The Cascade By Diagram: Author Styles Absent. User Styles Absent.



7.2. The Cascade Processing Author Styles Into An Author Styles List And User Styles Into A User Styles List

For each [Web page](#), the [Cascade](#) processes [author styles](#), if present, into an [author styles list](#), and [user styles](#), if present, into a [user styles list](#). As described in [The Cascade's Rules For Resolving Style Conflicts \(below\)](#), the order of the [styles](#) in the author styles list and user styles list plays a role in how the Cascade resolves [style conflicts](#). Therefore, it is important to understand how the Cascade processes author styles into an author styles list and user styles into a user styles list.

For the Cascade processing author styles into an author styles list, think of the Cascade as creating a giant [embedded style sheet](#) for each Web page, replacing any links to [external style sheets](#) with the external style sheets, themselves, and replacing any `@import` statements with the external style sheets to be imported, themselves. For the Cascade processing user styles into a user styles list, think of the Cascade as creating a giant external style sheet, replacing any `@import` statements with the external style sheets to be imported, themselves.

Suppose, for example, that a Web page has the following [source code](#):

Note: For this example, using actual [style rules](#) is superfluous and could be confusing. Therefore, generic style rule names (e.g., style rule a) are used instead of actual style rules.

```

<html>
<head>
<title></title>
<link rel="stylesheet" type="text/css" href="extOne.css" />
<style type="text/css">
@import "impTwo.css";
style rule k
style rule l
style rule m
</style>
  
```

```

<link rel="stylesheet" type="text/css" href="extTwo.css" />
</head>
<body>
...content here...
</body>
</html>

```

And suppose that the external style sheets, extOne.css and extTwo.css, and imported style sheets, impOne.css, impTwo.css, and impThree.css, are:

extOne.css

```

@import "impOne.css";
style rule d
style rule e

```

extTwo.css

```

style rule n
style rule o
style rule p

```

impOne.css

```

style rule a
style rule b
style rule c

```

impTwo.css

```

@import "impThree.css";
style rule i
style rule j

```

impThree.css

```

style rule f
style rule g
style rule h

```

To process the author styles into an author styles list, parse the Web page's source code and replace the links to the external style sheets with the external style sheets, themselves, and replace the @import statements with the external style sheets to be imported, themselves, as shown below:

```

<html>
<head>
<title></title>
                                     /style rule a
                                     /@import "impOne.css";( style rule b
                                     /
                                     \style rule c
<link rel="stylesheet" type="text/css" href="extOne.css" />( style rule d
                                     \ style rule e
<style type="text/css">
                                     /style rule f
                                     /@import "impThree.css";( style rule g
                                     /
                                     \style rule h
@import "impTwo.css";( style rule i
                       \ style rule j
style rule k
style rule l
style rule m
</style>
                                     /style rule n
<link rel="stylesheet" type="text/css" href="extTwo.css" />( style rule o
                                     \style rule p
</head>
<body>

```

```
...content here...
</body>
</html>
```

Extracting the author styles line by line from top to bottom, the author styles list is:

```
style rule a
style rule b
style rule c
style rule d
style rule e
style rule f
style rule g
style rule h
style rule i
style rule j
style rule k
style rule l
style rule m
style rule n
style rule o
style rule p
```

Note: In this example, the embedded style sheet is preceded by an external style sheet and is followed by an external style sheet. Therefore, in this example's author styles list, the style rules of the embedded style sheet are listed between the style rules of the preceding and following external style sheets.

7.3. The Cascade Merging The User Agent Default Styles, Author Styles List, User Styles List, And The Definition Of A Style Conflict

For each Web page, the Cascade merges (i.e., combines) the user agent default styles, author styles list, if present, and user styles list, if present. That the Cascade merges the user agent default styles, author styles list, if present, and user styles list, if present: 1.) allows user agent default styles, author styles, and user styles to simultaneously style content; and 2.) allows author styles and user styles to be partial styles (i.e., author styles and user styles need only assign the styles to add to, or change from, the user agent default styles).

When merging the user agent default styles, author styles list, if present, and user styles list, if present, the Cascade also checks for and resolves style conflicts. A style conflict is two or more styles attached to content in the same Web page having the same selector, the same property, but different values. In other words, style conflicts occur at the level of a style, not at the level of a style rule or style sheet. Style conflicts can occur: 1.) between the user agent default styles, and/or author styles list, and/or user styles list; and 2.) within the author styles list and/or user styles list. Style conflicts can cause styles to not be applied to content as expected. Therefore, it is important to clearly understand the definition of a style conflict.

7.3.1. The Definition Of A Style Conflict: Styles With The Same Property, The Same Value, But Different Selectors. No Style Conflict.

Consider, for example, the following style rules:

```
h1 { font-family: arial; margin-left: 1em }
h2 { font-family: arial; margin-left: 1em }
```

There are four styles above; 2 x font-family; arial, and 2 x margin-left: 1em. The two font-family; arial styles have the same property, the same value, but different selectors. The two margin-left: 1em styles have the same property, the same value, but different selectors. Styles with the same property, the same value, but different selectors do not have a style conflict.

7.3.2. The Definition Of A Style Conflict: Styles With The Same Selector, But Different Properties. No Style Conflict.

Consider, for example, the following style rules:

```
body { font-family: "times new roman"; font-size: 16px }
body { background-color: #ffffff }
body { color: #000000 }
```

There are four styles above; 1.) font-family: "times new roman", 2.) font-size: 16px, 3.) background-color: #ffffff, and 4.) color: #000000. The styles have the same selector, but different properties. Styles with the same selector, but different properties do not have a style conflict. Moreover, their declarations can be grouped into a semicolon separated declaration block as follows:

```
body { font-family: "times new roman"; font-size: 16px; background-color: #ffffff; color: #000000 }
```

7.3.3. The Definition Of A Style Conflict: Styles With The Same Selector, The Same Property, And The Same Value. No Style Conflict.

Consider, for example, the following style rules:

```
a:visited { background-color: transparent; color: #000000 }
a:visited { background-color: transparent; color: #000000 }
```

There are four styles above; 2 x background-color: transparent, and 2 x color: #000000. The two background-color: transparent styles have the same selector, the same property, and the same value. The two color: #000000 styles have the same selector, the same property, and the same value. Styles with the same selector, the same property, and the same value do not have a style conflict. Rather, the styles are in duplicate and their declarations can be grouped into a semicolon separated declaration block, eliminating the duplicate styles, as follows:

```
a:visited { background-color: transparent; color: #000000 }
```

7.3.4. The Definition Of A Style Conflict: Styles With The Same Selector, The Same Property, But Different Values. Style Conflict.

Consider, for example, the following style rules:

```
.special ( font-family: tahoma; color: #000000; font-size: 26px; border: 1px #000000 solid }
.special ( font-family: verdana; color: #000000; border-width: 2px }
.special ( font-family: century; font-size: 2em; border-color: #0000ff }
.special ( font-family: courier; color: #ffffff; font-size: 2em; border-style: dotted }
```

There are sixteen styles above; 4 x font-family, 3 x color, 3 x font-size, 2 x border-width, 2 x border-color, and 2 x border-style. The four font-family styles have the same selector, the same property, but four different values. The three color styles have the same selector, the same property, but two different values. The three font-size styles have the same selector, the same property, but two different values. The two border-width styles have the same selector, the same property, but two different values. The two border-color styles have the same selector, the same property, but two different values. The two border-style styles have the same selector, the same property, but two different values. Styles with the same selector, the same property, but different values (regardless how many different values) have a style conflict. Moreover, until the Cascade resolves the style conflicts, the .special style rule is essentially as follows:

```
.special ( font-family: ???; color: ???; font-size: ???; border: ??? ??? ??? }
```

7.4. The Cascade's Rules For Resolving Style Conflicts

The Cascade resolves style conflicts. To resolve a style conflict is to assign a style, from those attached to content in the same Web page having the same selector, the same property, but different values, to a selector. In other words, just as style conflicts occur at the level of a style, not at the level of a style rule or style sheet, style conflicts are resolved at the level of a style, not at the level of a style rule or style sheet. The styles that the Cascade assigns to selectors to resolve style conflicts are said to override (i.e., to overwrite, to suppress, to prevail over, to win over) the styles not assigned.

The W3C CSS2 Recommendation defines the following rules for the Cascade to resolve style conflicts:

Note: Line-through (i.e., ~~line-through~~) below designates advanced and/or infrequently used aspects of the Cascade's rules for resolving style conflicts that are beyond the scope of this page.

1. *"Find all declarations that apply to the element and property in question, ~~for the target media type~~. Declarations apply if ~~the associated selector matches the element in question~~." - W3C CSS2 Recommendation*
Translation: Process author styles, if present, into an author styles list, and user styles, if present, into a user styles list. Then merge the user agent default styles, author styles list, if present, and user styles list, if present, and find the style conflicts.
2. *"The primary sort of the declarations is by weight and origin: for normal declarations, author style sheets override user style sheets which override the default style sheet. For 'important' declarations, user style sheets override author style sheets which override the default style sheet. 'important' declaration override normal declarations. An imported style*

sheet has the same origin as the style sheet that imported it." - W3C CSS2 Recommendation

Translation: First try to resolve style conflicts based on the weight and origin of the conflicting styles, where weight means whether a style is marked !important or is normal (i.e., non-!important), and origin means whether a style is a user agent default style, author style, or user style. For normal styles, author styles override user styles which override user agent default styles. For !important styles, user styles override author styles which override user agent default styles. Regardless of origin, !important styles override normal styles. The origin of the styles of an imported style sheet is the same as the origin of the style sheet with the @import statement.

Note: That author styles can override user styles and, conversely, user styles can override author styles, defines author/user balance. The combination of marking styles !important, and the Cascade's rule for resolving style conflicts based on the weight of the conflicting styles, provides the mechanism for author/user balance.

3. ~~"The secondary sort is by specificity of selector: more specific selectors will override more general ones. Pseudo-elements and pseudo-classes are counted as normal elements and classes, respectively."~~ - W3C CSS2 Recommendation
4. ~~"Finally, sort by order specified: if two rules have the same weight, origin and specificity, the latter specified wins. Rules in imported style sheets are considered to be before any rules in the style sheet itself."~~ - W3C CSS2 Recommendation
Translation: If conflicting styles have the same weight and origin, and, therefore, cannot be resolved by rule 2 above, then the styles listed later in the author styles list override the styles listed earlier in the author styles list, and the styles listed later in the user styles list override the styles listed earlier in the user styles list. Reflecting that @import statements function only if they are listed at the top of a style sheet, and that @import statements indicate where the imported style sheets are to be inserted, which is in the place of the @import statements, themselves, the styles of an imported style sheet are listed earlier than the styles of the style sheet with the @import statement.

7.5. The Firefox 3.5 Cascade In Action

For The Firefox 3.5 Cascade In Action example that follows: 1.) the user agent is Firefox 3.5; 2.) author styles are present and include an embedded style sheet and an external style sheet, but user styles are absent; and 3.) the selectors are limited to the elements, body, h1, p, and a, the pseudo-classes, :link, :visited, and :hover, and a couple classNames.

7.5.1. Approximating The Firefox 3.5 User Agent Default Styles

The style sheet built into Firefox 3.5 appears to be, or to be based on, the file, C:\Program Files\Mozilla Firefox\res\html.css. Therefore, for the selectors in this example, the user agent default styles from the Firefox 3.5 built in style sheet are:

```
body { display: block; margin: 8px }
h1 { display: block; font-size: 2em; font-weight: bold; margin: .67em 0 }
p { display: block; margin: 1em 0 }
```

Like most user agents, Firefox 3.5 acquires additional user agent default styles from its Options/Preferences. In Firefox 3.5, click Tools | Options | Content. The Fonts & Colors fieldset appears. For the selectors in this example, the user agent default styles from the Firefox 3.5 Options/Preferences are:

```
body { font-family: "times new roman"; font-size: 16px; background-color: #ffffff; color: #000000 }
a { text-decoration: underline }
:link { color: #0000ff }
:visited { color: #800080 }
```

Merging the user agent default styles from the Firefox 3.5 built-in style sheet and the Firefox 3.5 Options/Preferences results in the following approximation of the Firefox 3.5 user agent default styles for the selectors in this example:

```
body { display: block; margin: 8px; font-family: "times new roman"; font-size: 16px; background-color: #ffffff; color: #000000 }
h1 { display: block; font-size: 2em; font-weight: bold; margin: .67em 0 }
p { display: block; margin: 1em 0 }
a { text-decoration: underline }
:link { color: #0000ff }
:visited { color: #800080 }
```

Note: For additional information on user agent default styles, see User Agent Default Styles (above).

7.5.2. The Firefox 3.5 Cascade Processing Author Styles Into An Author Styles List

For each Web page, the Cascade processes author styles, if present, into an author styles list, and user styles, if present, into a user styles list. In this example, author styles are present and include an embedded style sheet and an external style sheet, but

user styles are absent. Therefore, in this example, the Firefox 3.5 Cascade processes author styles into an author styles list, but no user styles list is generated.

Suppose that the source code for the Web page in this example is as follows:

```
<html>
<head>
<title></title>
<style type="text/css">
body { font-family: arial, verdana, "times new roman"; font-size: 12px; background-color: #eeeeee }
h1 { text-decoration: underline }
p { margin-left: 1em }
a { font-style: italic; background-color: #ffffff; text-decoration: underline }
a:link { color: #ff0000 }
a:visited { color: #a9a9a9 }
.special { font-size: 12px; font-family: "times new roman"; border: 2px #0000ff solid; padding: 0px 6px; margin-right: 10% }
</style>
<link rel="stylesheet" type="text/css" href="cascade_in_action_external_style_sheet.css" />
</head>
<body>
<div class="special">div .special</div>
<h1>Heading 1</h1>
<p>paragraph</p>
<a href="cascade_in_action_dummy_web_document1.html">Link 1</a><br />
<a href="cascade_in_action_dummy_web_document2.html">Link 2</a><br />
<a href="cascade_in_action_dummy_web_document3.html">Link 3</a><br />
<h1>Heading 1</h1>
<p>paragraph</p>
<p class="special">paragraph .special</p>
<p class="green">paragraph .green</p>
</body>
</html>
```

And suppose that the external style sheet in this example, `cascade_in_action_external_style_sheet.css`, is:

```
body { font-family: tahoma, "myriad web"; font-size: 20px }
h1 { font-style: italic }
a { background-color: transparent }
a:hover { color: #0000ff }
.special { font-size: 14px; background-color: #ffffff; padding-top: 6px; padding-bottom: 6px }
.green { color: #008000 }
```

The Firefox 3.5 Cascade processes the author styles in this example (i.e., the embedded style sheet and external style sheet above) into the following author styles list:

```
body { font-family: arial, verdana, "times new roman"; font-size: 12px; background-color: #eeeeee }
h1 { text-decoration: underline }
p { margin-left: 1em }
a { font-style: italic; background-color: #ffffff; text-decoration: underline }
a:link { color: #ff0000 }
a:visited { color: #a9a9a9 }
.special { font-size: 12px; font-family: "times new roman"; border: 2px #0000ff solid; padding: 0px 6px; margin-right: 10% }
body { font-family: tahoma, "myriad web"; font-size: 20px }
h1 { font-style: italic }
a { background-color: transparent }
a:hover { color: #0000ff }
.special { font-size: 14px; background-color: #ffffff; padding-top: 6px; padding-bottom: 6px }
.green { color: #008000 }
```

Note: For additional information on the Cascade processing author styles into an author styles list, see The Cascade Processing Author Styles Into An Author Styles List And User Styles Into A User Styles List (above).

7.5.3. The Firefox 3.5 Cascade Merging The Firefox 3.5 User Agent Default Styles, Author Styles List, And Resolving Style Conflicts

For each [Web page](#), the [Cascade](#) merges the [user agent default styles](#), [author styles list](#), if present, and [user styles list](#), if present, and resolves [style conflicts](#). In this example, an author styles list is present, but a user styles list is absent. Therefore, in this example, the Firefox 3.5 Cascade merges the Firefox 3.5 user agent default styles, author styles list, and resolves style conflicts as follows:

```
Author latter: body { font-family: tahoma, "myriad web"; font-size: 20px }
Author former: body { font-family: arial, verdana, "times new roman"; font-size: 12px; background-color: #eeeeee }
Firefox 3.5 user agent: body { display: block; margin: 8px; font-family: "times new roman"; font-size: 16px; background-color: #ffffff; color: #000000 }
Merged/resolved: body { font-family: tahoma, "myriad web"; font-size: 20px; background-color: #eeeeee; display: block; margin: 8px; color: #000000 }

Author latter: h1 { font-style: italic }
Author former: h1 { text-decoration: underline }
Firefox 3.5 user agent: h1 { display: block; font-size: 2em; font-weight: bold; margin: .67em 0 }
Merged/resolved: h1 { font-style: italic; text-decoration: underline; display: block; font-size: 2em; font-weight: bold; margin: .67em 0 }

Author: p { margin-left: 1em }
Firefox 3.5 user agent: p { display: block; margin: 1em 0 }
Merged/resolved: p { margin: 1em 0em 1em 1em; display: block }

Author latter: a { background-color: transparent }
Author former: a { font-style: italic; background-color: #ffffff; text-decoration: underline }
Firefox 3.5 user agent: a { text-decoration: underline }
Merged/resolved: a { background-color: transparent; font-style: italic; text-decoration: underline }

Author: a:link { color: #ff0000 }
Firefox 3.5 user agent: :link { color: #0000ff }
Merged/resolved: a:link { color: #ff0000 }

Author: a:visited { color: #a9a9a9 }
Firefox 3.5 user agent: :visited { color: #800080 }
Merged/resolved: a:visited { color: #a9a9a9 }

Author latter: .special { font-size: 14px; background-color: #ffffff; padding-top: 6px; padding-bottom: 6px }
Author former: .special { font-size: 12px; font-family: "times new roman"; border: 2px #0000ff solid; padding: 0px 6px; margin-right: 10% }
Firefox 3.5 user agent: .special does not exist
Merged/resolved: .special { font-size: 14px; background-color: #ffffff; padding: 6px; font-family: "times new roman"; border: 2px #0000ff solid; margin-right: 10% }

Author: a:hover { color: #0000ff }
Firefox 3.5 user agent: :hover does not exist
Merged/resolved: a:hover { color: #0000ff }

Author: .green { color: #008000 }
Firefox 3.5 user agent: .green does not exist
Merged/resolved: .green { color: #008000 }
```

Note: For additional information on:

- the [Cascade](#) merging the [user agent default styles](#), [author styles list](#), [user styles list](#), and the definition of a [style conflict](#), see [The Cascade Merging The User Agent Default Styles, Author Styles List, User Styles List, And The Definition Of A Style Conflict \(above\)](#).
- the Cascade's rules for resolving style conflicts, see [The Cascade's Rules For Resolving Style Conflicts \(above\)](#).

7.5.4. The Firefox 3.5 Cascade Creating The Master Style Sheet

For each [Web page](#), the [Cascade](#) creates a master [style sheet](#) whose [styles](#) the [user agent's rendering engine](#) applies to [content](#) in the [Web Browser window](#). In this example, the Firefox 3.5 Cascade creates the following master style sheet:

```
body { font-family: tahoma, "myriad web"; font-size: 20px; background-color: #eeeeee; display: block; margin: 8px; color: #000000 }
h1 { font-style: italic; text-decoration: underline; display: block; font-size: 2em; font-weight: bold; margin: .67em 0 }
```

```
p { margin: 1em 0em 1em 1em; display: block }
a { background-color: transparent; font-style: italic; text-decoration: underline }
a:link { color: #ff0000 }
a:visited { color: #a9a9a9 }
.special { font-size: 14px; background-color: #ffffff; padding: 6px; font-family: "times new roman"; border: 2px #0000ff solid;
margin-right: 10% }
a:hover { color: #0000ff }
.green { color: #008000 }
```

Note: As described in [Style Rule Syntax And Conventions \(above\)](#), the order of [style rules](#) in a [style sheet](#) does not matter except for the order of style rules with [pseudo-classes](#) as [selectors](#). In the master style sheet, the order of the style rules with pseudo-classes as selectors is the same as their order in the [author styles list](#) and/or [user styles list](#).

7.5.5. The Firefox 3.5 Rendering Engine Applying The Styles Of The Master Style Sheet To Content: Proof That This Is The Way CSS Works

For each [Web page](#), the [Cascade](#) passes the master [style sheet](#) to the [user agent's rendering engine](#), which applies the [styles](#) to the Web page's [content](#) in the [Web Browser window](#). In this example, the Firefox 3.5 Cascade passes the master style sheet to the Firefox 3.5 rendering engine, which applies the styles to the Web page's content in the Firefox 3.5 Web Browser window.

If this Web page's understanding of [CSS](#), as represented by this example's master style sheet, is correct, particularly: 1.) that user agents have a style sheet built into them whose styles; a.) define and provide the [structure](#) types associated with [HTML/XHTML elements](#), and b.) are merged with [author styles](#) and [user styles](#) in styling content; 2.) that [style conflicts](#) occur at, and are resolved at, the level of a style; and 3.) that the styles of an [external style sheet](#) can override the styles of an [embedded style sheet](#), then the following two Web pages should appear identical in Firefox 3.5:

[Web Page Styled Via The Original Embedded Style Sheet And External Style Sheet \(stevestechresource.com\).](#)

[Web Page Styled Via The Firefox 3.5 Cascade In Action Master Style Sheet \(stevestechresource.com\).](#)

Important Note: Since the [user agent](#) in this example is Firefox 3.5, and, therefore, the master [style sheet](#) includes the Firefox 3.5 user agent default styles, the appearance of the two [Web pages](#) above should only be compared in Firefox 3.5, or in a Mozilla-based Web Browsers whose version of the [Cascade](#) and the Gecko [rendering engine](#) is identical to that of Firefox 3.5. In other words, if the appearance of the two Web pages above are compared in, for example, Microsoft Internet Explorer, the differences in their appearance reflects the differences between the Firefox 3.5 and Microsoft Internet Explorer user agent default styles, not on this Web page's understanding of [CSS](#).

8. Additional Reading

- [Cascading Style Sheets, Level 1: W3C Recommendation 17 Dec 1996 \(w3.org\)](#)
- [Cascading Style Sheets, Level 1: W3C Recommendation 17 Dec 1996, Revised 11 Apr 2008 \(w3.org\)](#)
- [Cascading Style Sheets, Level 2, CSS2 Specification: W3C Recommendation 12-May-1998 \(w3.org\)](#)
- [Cascading Style Sheets, Level 2, CSS2 Specification: W3C Recommendation 12-May-1998 \(Revised 11 April 2008\) \(w3.org\)](#)
- [Cascading Style Sheets Level 2 Revision 1 \(CSS 2.1\) Specification: W3C Recommendation 07 June 2011 \(w3.org\)](#)
- [Cascading Style Sheets: Thesis Submitted For The Degree Of Doctor Philosophy, Faculty Of Mathematics And Natural Sciences, University Of Oslo, Norway, 2005 \(people.opera.com\) \(Hakon Wium Lie.\) \(Comprehensive and verbose review of the history of CSS by one of its inventors.\)](#)
- [Sample Style Sheet For HTML 2.0 \(w3.org\)](#)
- [Default Style Sheet For HTML 4.0 \(w3.org\)](#)
- [W3C CSS Validation Service: Check Cascading Style Sheets \(CSS\) And \(X\)HTML Documents With Style Sheets \(jigsaw.w3.org\)](#)
- [Hypertext Markup Language - 2.0: Request For Comment 1866 November 1995 \(ftp.ics.uci.edu\) \(Tim Berners-Lee.\)](#)
- [HTML 3.2 Reference Specification: W3C Recommendation 14-Jan-1997 \(w3.org\)](#)
- [HTML 4.0 Specification: W3C Recommendation 18-Dec-1997 \(w3.org\)](#)
- [HTML 4.0 Specification: W3C Recommendation, Revised On 24-Apr-1998 \(w3.org\)](#)
- [HTML 4.01 Specification: W3C Recommendation 24 December 1999 \(w3.org\)](#)
- [XHTML 1.0 The Extensible HyperText Markup Language \(Second Edition\): A Reformulation Of HTML 4 In XML 1.0: W3C Recommendation 26 January 2000, Revised 1 August 2002 \(w3.org\)](#)
- [XHTML 1.1: Module-Based XHTML: W3C Recommendation 31 May 2001 \(w3.org\)](#)
- [Document Object Model \(DOM\) Technical Reports \(w3.org\)](#)

Steve's Tech Resource

The Web Development, Internet, Software, Hardware, and Multimedia Resource



Copyright © 2000-2012 Steve's Tech Resource