Cascading Style Sheets

Luca Bompani

20 novembre 2000

Introduction

- CSS syntax
- Browsers
- XML Conversion Tools
- Document Storage Systems
- DTD Development Tools

Introduction

CSS2 is a style sheet language that allows authors and users to attach style to structured documents (e.g., HTML documents and XML applications).
By separating the presentation style of documents from the content of documents, CSS2 simplifies Web authoring and site maintenance.
CSS2 supports media-specific style sheets so that authors may tailor the presentation of their documents to visual browsers, aural devices, printers, braille devices, handheld devices, etc.
CSS also supports content positioning, downloadable fonts, table layout, features for internationalization, automatic counters and numbering, and some properties related to user interface.

Example (1)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
 <HEAD>
  <TITLE>Bach's home page</TITLE>
  <STYLE type="text/css">
   BODY{color: red}
   H1{color: blue}
  </STYLE>
 </HEAD>
 <BODY>
  <H1>Bach's home page</H1>
   <P>Johann Sebastian Bach was a prolific composer.
 </BODY>
</HTML>
```


Example (2)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
 <HEAD>
  <TITLE>Bach's home page</TITLE>
  <LINK rel="stylesheet" href="bach.css" type="text/css">
 </HEAD>
 <BODY>
  <H1>Bach s home page</H1>
  <P>Johann Sebastian Bach was a prolific composer.
 </BODY>
</HTML>
```


Syntax

```
stylesheet: [CDO|CDC|S|statement]*;
statement: ruleset|at-rule;
at-rule: ATKEYWORD S* any* [block|;S*];
block: {S*[any|block| ATKEYWORD S*|;]* }S*;
ruleset: selector?{S* declaration?[;S*
         declaration? ]*} S*;
selector: any+;
declaration: property : S* value;
property: IDENT S*;
value: [any|block|ATKEYWORD S*]+;
any: [IDENT|NUMBER|PERCENTAGE|DIMENSION|STRING|DELIM|
      URI|HASH|UNICODE-RANGE|INCLUDES|FUNCTION|
      DASHMATCH|(any*)|[any*]] S*;


CDO  : '<!--'
CDC: '-->'
ATKEYWORD: '@'
```


At-rules

At-rules start with an at-keyword, an @ character followed by an identifier (for example, @import, @page).
An at-rule consists of everything up to and including the next semicolon (;) or the next block.
CSS2 user agents must ignore any @import rule that occurs inside a block or that doesn t precede all rule sets.

```
@import "subs.css";
@media print {
 @import "print-main.css";
 BODY {font-size: 10pt}
}
H1 {color: blue}
```


Blocks

A block starts with a left curly brace ({) and ends with the matching right curly brace (}). In between there may be any characters, except that parentheses (( )), brackets ([ ]) and braces ({ }) must always occur in matching pairs and may be nested.
Single ( ) and double quotes (") must also occur in matching pairs, and characters between them are parsed as a string.

```
{ causta: "}" + ({7} *  \  ) }
```


Rule set

A rule set consists of a `selector` followed by a `declaration block`.
A **declaration-block** starts with a left curly brace ({) and ends with the matching right curly brace (}). In between there must be a list of zero or more semicolon-separated (;) declarations.
The **selector** consists of everything up to (but not including) the first left curly brace ({). A selector always goes together with a {}-block.

```
H1,H2{color: green }
H3,H4,H5{color: red }
H6{color: black }
```


Declaration

A declaration is either empty or consists of a property, followed by a colon (:), followed by a value.
Around each of these there may be whitespace.
Because of the way selectors work, multiple declarations for the same selector may be organized into semicolon (;) separated groups.

```
H1 { font-weight: bold }
H1 { font-size: 12pt }
H1 { font-family: Helvetica }

H1{
 font-weight: bold;
 font-size: 12pt;
 font-family: Helvetica;
}
```


Property

A property is an identifier. Any character may occur in the value, but parentheses ("( )"), brackets ("[ ]"), braces ("{ }"), single quotes ( ) and double quotes (") must come in matching pairs, and semicolons not in strings must be escaped.
Parentheses, brackets, and braces may be nested. Inside the quotes, characters are parsed as a string.

```
H1 { color: red; }
P { color: blue; font-variant: small-caps }
EM EM { font-style: normal }
```


Values (1)

Lengths

The format of a length value is a number immediately followed by a unit identifier. There are two types of length units: relative and absolute.

Relative units are:

- `em`: the font-size of the relevant font
- `ex`: the x-height of the relevant font
- `px`: pixels, relative to the viewing device

The absolute units are:

- `in`: inches -- 1 inch is equal to 2.54 centimeters.
- `cm`: centimeters
- `mm`: millimeters
- `pt`: points -- 1 point is equal to 1/72th of an inch.
- `pc`: picas -- 1 pica is equal to 12 points.

Values (2)

Integers and real numbers

Real numbers and integers are specified in decimal notation only.

Percentages

The format of a percentage value is a number immediately followed by %. Percentage values are always relative to another value. Each property that allows percentages also defines the value to which the percentage refers. The value may be that of another property for the same element, a property for an ancestor element, or a value of the formatting context.

```
P{font-size: 10pt}
P{line-height: 120%} /*120% of font-size  */
```


Values (3)

URL + URN = URI

URLs provide the address of a resource on the Web. An expected new way of identifying resources is called URN. Together they are called URIs. The format of a URI value is `url(` followed by the URI itself, followed by `)`.

```
BODY{background:url("http://www.bg.com/pinkish.gif")}
```

Counters

Counters are denoted by identifiers. To refer to the value of a counter, the notation used is `counter(<id>)` or `counter(<id>, <list-style-type>)`. The default style is `decimal` .

```
P{counter-increment: par-num}
H1{counter-reset: par-num}
P:before{content: counter(par-num,upper-roman) ". "}
```


Values(4)

Colors

A color is either a keyword or a numerical RGB specification. The list of keyword color names is: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. The RGB color model is used in numerical color specifications.

```
EM { color: #ff0000 }
EM { color: rgb(255,0,0) }
EM { color: rgb(100%, 0%, 0%) }
```

Angles

The format of angle values is a number followed by an angle unit identifier. Angle unit identifiers are:

- `deg`: degrees
- `grad`: grads
- `rad`: radians

Values(5)

Times

Time values are used with aural style sheets. Their format is a number immediately followed by a time unit identifier. Time unit identifiers are:

- ms: milliseconds
- s: seconds

Frequencies

Frequency values used with aural cascading style sheets. Their format is a number immediately followed by a frequency unit identifier. Frequency unit identifiers are:

Hz: Hertz

kHz: kilo Hertz

Strings

Strings can either be written with double or single quotes.

Selector (1)

In CSS, pattern matching rules determine which style rules apply to elements in the document tree. These patterns, called selectors, may range from simple element names to rich contextual patterns.
If all conditions in the pattern are true for a certain element, the selector matches the element.
The case-sensitivity of document language element names in selectors depends on the document language. For example, in HTML, element names are case-insensitive, but in XML they are case-sensitive.

Selector(2)

`*` Matches any element
`E` Matches any E element
`E F` Matches any F element that is a descendant of an E element.
`E > F` Matches any F element that is a child of an element E.
`E + F` Matches any F element immediately preceded by an element E.
`E:first-child` Matches element E when E is the first child of its parent.
`E:link E:visited` Matches element E if E is the source anchor of a hyperlink of which the target is not yet visited (:link) or visited (:visited).
`E:active E:hover E:focus` Matches E during certain user actions.
`E:lang(c)` Matches element of type E if it is in (human) language c (the document language specifies how language is determined).
`E[foo]` Matches any E element with the "foo" attribute set
`E[foo="warning"]` Matches any E element whose "foo" attribute value is exactly equal to "warning"
`E[foo~="warning"]` Matches any E element whose "foo" attribute value is a list of space-sepa-rated values, one of which is exactly equal to "warning".
`E[lang|="en"]` Matches any E element whose "lang" attribute has a hyphen-separated list of values beginning (from the left) with "en".
`DIV.warning` HTML only. The same as DIV[class~="warning"].
`E#myid` Matches any E element ID equal to "myid".

Inheritance

Some values are inherited by the children of an element in the document tree. Each property defines whether it is inherited or not.
Suppose there is an H1 element with an emphasizing element (EM) inside:

```
<H1>The headline <EM>is</EM> important!</H1>
```

If no color has been assigned to the EM element, the emphasized "is" will inherit the color of the parent element, so if H1 has the color blue, the EM element will likewise be in blue.
To set a "default" style property for a document, authors may set the property on the root of the document tree.
In HTML, for example, the HTML or BODY elements can serve this function.

The cascade (1)

Style sheets may have three different origins: author, user, and user agent.

The **author** specifies style sheets for a source document according to the conventions of the document language. For instance, in HTML, style sheets may be included in the document or linked externally.

The **user** may be able to specify style information for a particular document. For example, the user may specify a file that contains a style sheet or the user agent may provide an interface that generates a user style sheet (or behave as if it did).

Conforming **user agents** must apply a default style sheet (or behave as if they did) prior to all other style sheets for a document.

The cascade (2)

To find the value for an element/property combination, user agents must apply the following sorting order:

1. Find all declarations that apply to the element and property in question, for the target media type.

2. The primary sort of the declarations is by origin: for normal declarations, author style sheets override user style sheets which override the default style sheet.

3. The secondary sort is by specificity of selector: more specific selectors will override more general ones.

4. Finally, sort by order specified: if two rules have the same weight, origin and specificity, the latter specified wins.

Media type (1)

One of the most important features of style sheets is that they specify how a document is to be presented on different media: on the screen, on paper, with a speech synthesizer, with a braille device, etc.
Certain CSS properties are only designed for certain media.
On occasion, however, style sheets for different media types may share a property, but require different values for that property.
However, the two media are different enough to require different values for the common property; a document will typically need a larger font on a computer screen than on paper.

Media type (2)

There are currently two ways to specify media dependencies for style sheets:

Specify the target medium from a style sheet with the @media or @import at-rules:

```
@import url("loudvoice.css") aural;
@media print {/* style sheet for print goes here */}
```

Specify the target medium within the document language. For example, in HTML 4.0 ([HTML40]), the "media" attribute on the LINK element specifies the target media of an external style sheet:

```
<HTML>
 <HEAD>
  <LINK rel="stylesheet" type="text/css"
    media="print, handheld" href="foo.css">
 </HEAD>
 <BODY><P>The body  </BODY>
```


Media type(3)

Css media types are:
`all:` suitable for all devices
`aural:` intended for speech synthesizers
`braille:` intended for braille tactile feedback devices
`embossed:` intended for paged braille printers
`handheld:` intended for handheld devices
`print:` intended for paged, opaque material and for
`documents:` viewed on screen in print preview mode
`projection:` intended for projected presentations
`screen:` intended primarily for color computer screens
`tty:` intended for media using a fixed-pitch character grid, such as teletypes, terminals, or portable devices
`tv:` intended for television-type devices

References

Cascading Style Sheets -level 2 CSS2 Specification
W3C Recommendation 12-May-1998
`http://www.w3.org/TR/1998/REC-CSS2-19980512`