# Cascading Style Sheets (CSS)

## Web Architecture and Information Management [./]
## Spring 2009 — INFO 190-02 (CCN 42509)

Erik Wilde, UC Berkeley School of Information

**2009-02-04**









# Contents

## Abstract (2)

*Cascading Stylesheets (CSS)* have been designed as a language for better separating presentation-specific issues from the structuring of documents as provided by HTML. CSS uses a simple model of *selectors* and *declarations*. Selectors specify to which elements of a document a set of declarations (each being a value assigned to a property) apply; in addition there is a model of how property values are inherited and cascaded. The biggest limitation of CSS is that it cannot change the structure of the displayed document.







# Why CSS?

## Structure vs. Layout (4)

- HTML started as very simple layout-oriented structures
  - more layout control was introduced as attributes (align, color)
  - HTML became increasingly "polluted" by layout information
- CSS was introduced as a format for layout information
  - the HTML can be kept simple, containing only the structures
  - layout information can be reused by using separate CSS files
- CSS has been designed for HTML
  - it has been generalized to also cover XML
  - the HTML heritage is still very visible in CSS

## What's Still Missing? (5)

1. Restructuring content
   - CSS assigns formatting properties to elements
   - the document tree which is formatted cannot be restructured
   - parts can be *ignored* or new parts can be *inserted*
2. Interpreting content
   - img has a lot of special meanings attached
   - form elements have special semantics (such as creating input fields)

- Scripting can be used for implementing some of these issues
- CSS is simply for more sophisticated styling of HTML pages







# How CSS Works

## CSS in Action (7)

```
<body id="css-zen-garden">
<div id="container">
 <div id="intro">
  <div id="pageHeader">
   <h1><span>css Zen Garden</span></h1>
   <h2><span>The Beauty of <acronym title="Cascading Style
Sheets">CSS</acronym> Design</span></h2>
  </div>
  <div id="quickSummary">
   <p class="p1"><span>A demonstration of what can be accomplished
visually through <acronym title="Cascading Style
Sheets">CSS</acronym>-based design. Select any style sheet from the
list to load it into this page.</span></p>
   <p class="p2"><span>Download the sample <a href="/zengarden-
sample.html" title="This page's source HTML code, not to be
modified.">html file</a> and <a href="/zengarden-sample.css"
title="This page's sample CSS, the file you may modify.">css file</a>
</span></p>
  </div>
  <div id="preamble">
   <h3><span>The Road to Enlightenment</span></h3>
   <p class="p1"><span>Littering a dark and dreary road lay the past
relics of browser-specific tags, incompatible <acronym
title="Document Object Model">DOM</acronym>s, and broken <acronym
title="Cascading Style Sheets">CSS</acronym> support.</span></p>
```

# HTML and CSS (8)

- CSS specifies how HTML elements are formatted
  1. formatting can be attached to every element (redundant inside document)
  2. formatting can be included in document (redundant across documents)
  3. separate CSS files describe the formatting (best reuse)
- Any combination of these methods is possible

```
<head>
 <title>CSS Usage</title>
 <link rel="stylesheet" href="http://dret.net/dretnet.css"
type="text/css"/>
 <style type="text/css"> li { color : red } </style>
</head>
<body>
 <p>some text in a paragraph..</p>
 <ol>
  <li>an ordered list's first item</li>
  <li style=" text-decoration : underline ">and the second one</li>
 </ol>
```





# Formatting Model (9)

- Properties [Properties (1)] are central to the CSS formatting model
  1. create a document tree
  2. identify the media type (e.g., screen or print)
  3. retrieve all stylesheets required for the media type
  4. assign values to all properties in the document tree
  5. generate a *formatting structure* (a different tree)
  6. render the formatting structure on the target medium
- Properties control the rendering of elements
- Styling in CSS means assigning values to properties

# CSS Strategies

## Use Classes & Containers (11)

- CSS code should never show up in your HTML
- Classes should reflect content or formatting
- Containers [Advanced HTML; All-Purpose Elements (1)] can restrict styles to a context
- Context can be nested
  - orthogonal concepts should be represented as nested classes
  - for example, pages for "staff" and "faculty" and "current" and "past" as classification
  - different levels of formatting sophistication can be implemented with CSS only
- Avoid redundant CSS code
  - Whenever appropriate, inherit properties
  - for invisible links use a { color : inherit ; }







# Properties

## Formatting Instructions (13)

- Properties define how elements are formatted
  - they define a specific facet of formatting
  - they may have interdependencies with other properties
  - they can be assigned explicitly
  - they may be defined through Cascading [Cascading (1)] or Inheritance [Inheritance (1)]
- A property has a name and is used in a name/value-pair
  - the name identifies the property that is being set
  - the value space depends on the property
  - some properties accept complex values
  - sets of values: p { font : bold italic large Palatino }
  - sequences of values: p { font-family : "Segoe UI", verdana, helvetica, arial, sans-serif }
- Property specifications can be grouped
  - .thinboxed { border-width : 1px ; padding : 10px ; margin : 5px }

# CSS1 Properties

## Factoring out HTML (15)

- CSS1 was published in 1996 [http://www.w3.org/TR/REC-CSS1-961217] and revised in 1999 [http://www.w3.org/TR/1999/REC-CSS1-19990111]
- HTML suffered from too many attributes
  - layout information was specified as CSS
  - style attributes in HTML were marked as "deprecated"
- A small set of formatting features as CSS properties
  - font [http://www.w3.org/TR/REC-CSS1/#font-properties]: p { font : 80% sans-serif }
  - color and background [http://www.w3.org/TR/REC-CSS1/#color-and-background-properties]: body { background : url(logo.jpeg) right top }
  - text [http://www.w3.org/TR/REC-CSS1/#text-properties]: h1 { text-transform : uppercase }
  - box [http://www.w3.org/TR/REC-CSS1/#box-properties]: p.quote { border-style : solid dotted }
  - classification [http://www.w3.org/TR/REC-CSS1/#classification-properties]: img { display : none }







# CSS2 Properties

## CSS2 (17)

- CSS2 was published in 1998 [http://www.w3.org/TR/1998/REC-CSS2-19980512/] and is still being revised (CSS2[1]) [http://www.w3.org/TR/CSS21/]
- CSS2[1] is what you can expect from modern browsers
  - with IE (even IE7) being the exception
- CSS2 is a single and coherent specification
  - CSS3 [http://www.w3.org/TR/css3-roadmap/] is a jungle of concurrent module development
  - CSS3 will never be finished (some modules will, though)

# Generated Content (18)

- CSS1 had no way of adding information to the document
  - by using display, parts of the document could be ignored
- *Generated content* [http://www.w3.org/TR/CSS21/generate.html] allows content to come from the CSS
  - only possible with :before and :after *pseudo-elements*
  - static strings: p.abstract:before { content : "Abstract: " }
  - special effects like "quotes": q:before { content : open-quote }
  - counters: h1:before { content: "Chapter " counter(chapter) ". " ; counter-increment : chapter }
- Quotes can be defined as being language dependent
  - q:lang(en) { quotes : "'" "'" "'" "'" }
  - q:lang(no) { quotes : "«" "»" "'" "'" }









# Tables (19)

- CSS1 does not address table formatting
  - table layout still had to be done using HTML attributes
  - a lot of redundant code specifying cell alignment and borders
- CSS2 introduces tables on the CSS level

```
table    { display: table }
tr       { display: table-row }
thead    { display: table-header-group }
tbody    { display: table-row-group }
tfoot    { display: table-footer-group }
col      { display: table-column }
colgroup { display: table-column-group }
td, th   { display: table-cell }
caption  { display: table-caption }
```

## Fixed vs. Automatic Table Layout (20)

- HTML defines a complex table rendering algorithm
  - tables are rendered incrementally
  - table layout is determined by looking at the complete table

**Automatic**

| | | |
|---|---|---|
| col 1 row 1 | col 2 row 1 col 2 row 1 | col 3 row 1 col 3 row 1 col 3 row 1 |
| col 1 row 2 | col 2 row 2 col 2 row 2 | col 3 row 2 col 3 row 2 col 3 row 2 |

**Fixed**

| | | |
|---|---|---|
| col 1 row 1 | col 2 row 1 col 2 row 1 | col 3 row 1 col 3 row 1 col 3 row 1 |
| col 1 row 2 | col 2 row 2 col 2 row 2 | col 3 row 2 col 3 row 2 col 3 row 2 |







# Selectors

## Select and Style (22)

- Properties [Properties (1)] are applied to elements
  - properties can be directly applied in an element's style attribute
  - in all other cases, *selectors* are used to select the styled elements
- Selectors are good for reusable CSS code
  - identifying the most appropriate formatting classes is not easy
  - planning for CSS for a larger site is a difficult task
- CSS project management should separate selectors and properties
  1. selectors are about which things should be identified and styled
  2. properties are about how this styling is implemented

# CSS1 Selectors

## CSS for Dummies (24)

- Very small set of selectors
  - selecting elements by name: h1 { font-size : large }
  - selecting elements by their id: #author { font-weight : bold }
  - selecting elements by their class: .abstract { font-size : small }
  - combining these mechanisms: p.warning { color : red }
- Pseudo-classes and -elements allow interesting effects
  - a links have state: a:visited and a:active
  - selection without markup: p:first-letter and p:first-line







# CSS2 Selectors

## More Selectors (26)

- CSS1 Selectors [CSS1 Selectors (1)] are available
  - element name, id, class, and combinations of these
- CSS2 introduced many new selectors
  - descendants: ul li { font : italic }
  - children: ul > li { font : italic }
  - adjacent siblings: h1 + h2 { margin-top : 0.5em }
  - attribute matching: h1[lang=nl] { color : orange }
- CSS2 selectors are sufficient for most tasks
- Setting class attributes is very important

## CSS2 Pseudo Classes (27)

- CSS1's pseudo-elements [CSS1 Selectors (1)] are available
  - link states and first letter and line of content
- CSS2 adds more qualifications for elements
  - first child: p:first-child { text-indent : 0 }
  - dynamic behavior: a:hover { … } a:active { … } a:focus { … }
  - language: :lang(de) { quotes: '»' '«' '‹' '›' }
  - Generated Content [Generated Content (1)]: q:before { content : open-quote } q:after { content : close-quote }







# CSS Mechanics

## Cascading (29)

- Stylesheets may have three different origins
  1. *page authors* associate CSS with their pages
  2. *users* configure their browser to use some CSS
  3. *user agents (browsers)* have built-in CSS how to style content
- Conflicts must be resolved using the following algorithm
  1. find all matching declarations (matching media type and selector)
  2. sort according to importance (browser < user < author)
  3. same importance must be sorted by specificity (more specific selectors)
  4. finally, sort by order in which they were specified
- !important rules can influence the algorithm
  - they are interpreted in step 2 (sorting by importance)
  - browser < user < author < author(important) < user(important)

# Inheritance (30)

- Properties often are inherited by children
  - setting a table's color sets the color for all contents
  - without inheritance, CSS stylesheets would have to be very large
- Inheritance is mostly intuitive
  - in reality, it is a bit more complicated

1. *specified value:* what the property specified (Cascading [Cascading (1)], inheritance, or initial)
2. *computed value:* computed based on the environment (e.g., ex → px)
3. *used value:* converted to an absolute value (e.g., percentage widths)
4. *actual value:* specific for the environment (e.g., borders with pixel fractions)









# Structuring Stylesheets (31)

- Stylesheets may need to be structured
  - importing CSS code is supported: @import "/dretnet.css" ;
  - modules of CSS code can be reused in different contexts
- Stylesheets may be specific for a media type
  - *braille, embossed, handheld, print, projection, screen, speech, tty, tv*
  - specified in HTML: link rel="stylesheet" type="text/css" media="print" href="/print.css"
  - specified in CSS: @media print { … }
  - media-dependent import: @import "/print.css" print ;

# CSS Conclusions (32)

- Appropriate for HTML
  - Flexible selection of elements using Selectors [Selectors (1)]
  - Powerful formatting of elements using Properties [Properties (1)]
  - Interesting interface design with *pseudo-classes* and *-elements*
- Inappropriate for general publishing
  - documents often need to be restructured
  - XML → HTML+CSS is a popular Web publishing setup