The Latest in Cascading Style Sheets

Chris Lilley, W3C
chris@w3.org
www.w3.org/people/chris

Online Help Conference London 2000
Style Sheets needed

The web without style sheets

- HTML tag abuse
- Presentational and proprietary elements
- Unstructured and badly-formed documents common
- **FONT** tag to control color, size, font face
- Tables to fake margins
- Spacer GIFs
- Images of well-set type, for headings

Benefits of style sheets

- HTML and XML describes content and structure
- CSS describes presentation
- CSS makes pages:
  - lighter, fewer images of text
  - more accessible
  - more beautiful
  - more democratic: both users and authors can influence presentation
Relationship with HTML

- CSS is independent of HTML
- HTML-specific issues in "HTML and Style Sheets"
- CSS1 fully replaces HTML extensions
- HTML display possible without supplied stylesheet
  - 'Browser default' stylesheet for HTML
  - Unfortunately, hardcoded bugwards-compatible rendering

The Web with XML

- Free choice of document type
- No legacy presentational information
- Well formed documents mandatory
- DTD-less processing possible (and common)
- Presentation-less XML common
  - RDF
  - CDF
  - etc ...
  - *no default semantics or presentation*
Relationship with XML

- CSS is independent of XML
- Linking to stylesheets via PI
- No 'extensions' need replacing 😊
- No hardcoded rendering to get in the way
- XML display not possible without supplied stylesheet

CSS history

- **Sep 1994** initial CSS proposal
- **Dec 1996** CSS1 becomes W3C Recommendation
- **May 1998** CSS2 becomes W3C Recommendation
- **Apr 1999** SVG Working Draft
- **Aug 1999** SMIL Animation Working Draft
- **Dec 1999** DOM 2 CSS Object Model Candidate Recommendation
### CSS Implementations

<table>
<thead>
<tr>
<th>HTML&amp;CSS</th>
<th>XML&amp;CSS</th>
<th>SVG&amp;CSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla</td>
<td>Netscape 6</td>
<td>CSIRO SVG toolkit</td>
</tr>
<tr>
<td>MS IE4</td>
<td>Opera 4</td>
<td>IBM SVG viewer</td>
</tr>
<tr>
<td>Opera 4</td>
<td>DocZilla</td>
<td>Adobe SVG plugin</td>
</tr>
<tr>
<td>ICE 5</td>
<td>HP ChaiFarer</td>
<td>Adobe Illustrator</td>
</tr>
<tr>
<td></td>
<td>ICHITARO Ark</td>
<td>Corel Draw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koala SVG viewer</td>
</tr>
</tbody>
</table>

### CSS basics

- Simple, declarative style sheet language for structured documents
- Cascading
  - combines author/reader/browser style sheets
  - allows compact style sheets (B, I, PRE)
- Applies to HTML and XML
- Style can be: external, embedded, interleaved
- Selectors/properties/values

```css
P { text-indent: 3em }
```
Cascading Style Sheets

- Cascades the readers, authors and browsers style sheets together
- Formatting object tree nearly identical to document tree
- Selectors determine the elements to be styled
- Formatting properties are given values

```css
.slide > item { color: green }
```

Selectors

- Element name, class, ID
- Any other attribute
- Contextual
- Pseudo-classes, pseudo-elements

```css
h1 {color: red}
img.logo {float: right}
#xyz {font-variant: small-caps}
```

```css
h1 em {font-style: italic}
:visited {border: solid red}
P:first-line {font-variant: small-caps}
```
Properties

. font
. background/color
. margin/border/padding
. display, float, clear, white-space
. page breaking
. position, z-order, clip, overflow
. fill, stroke, gradient
. ..... *lots others*

Values

. length units: em, ex, px, in, cm, mm, %
. color: "red", #F00, #FF0000, rgb(255, 255, 43)
. font: "Courier New", monospace
. angles: 45deg, 0.27rad
. URL: url(http://my.url/goes/here)
Example of CSS

BODY {
  margin-left: 12%;
  margin-right: 7em;
  color: #000609;
  background: #fff;
  font-size: 40pt;
  font-family: tahoma, trebuchet, "gill sans",
              arial, helvetica, sans-serif;
}

Visual formatting model

- Box-oriented
- Supports left-to-right, right-to-left
- Scalable across sizes
Formatting model details

- Auto-generation of tables and table rows
- Addition of columns, headers, footers
- Being extended for vertical writing

Background images

- Background colors and images
  - on any element, not just BODY
- Control over positioning
- Control over horizontal and vertical repeat
- Partially transparent images compositing on background color
Media-specific style sheets

```css
BODY {
    color: black;
    background: white;
}
@media tv {
    BODY {
        color: white;
        background: black;
    }
}
```

Floating elements vs tables

- HTML tables commonly used for layout
- Floating element is more scalable
- Text flows to left or right of floated element (not just images)
CSS speeds up the Web

To our surprise, style sheets promise to be the biggest possibility of major network bandwidth improvements, whether deployed with HTTP/1.0 or HTTP/1.1, by significantly reducing the need for inlined images to provide graphic elements, and the resulting network traffic. Use of style sheets whenever possible will result in the greatest observed improvements in downloading new web pages, without sacrificing sophisticated graphics design.

Network Performance Effects of HTTP/1.1, CSS1, and PNG

CSS2 built on CSS1:

- CSS Positioning
- CSS Printing extensions
- ACSS: Aural Cascading Style Sheets
- Web Fonts
- Collected together into CSS2:
  - addressing based on any attribute
  - automatic numbering
  - cursors
  - table formatting
Font problems

Nearly the same font
  . Style sheet calls for Optima
  . Reader has ITC Legacy Sans installed
  . Looks nearly the same, can't be used
  . Reader has ATM installed
  . Font could be synthesized, given data

WebFont solution

Font descriptors
  . Information about the fonts
  . Stored in @font-face
  . Can include download information (url, formats)
  . Can include font and glyph metrics
  . Can include Unicode coverage hints
**Strengths**

- Extends, rather than replacing, CSS1 font model
- Draw on industry practice (Panose) and experience (Acrobat)
- Allows use of multiple fonts to build coverage (simplifies hinting)
- Font format agnostic (use content negotiation)
- Display technology agnostic (downloadable renderers possible)
- Three implementations

**Known limitations**

- No common format between implementations
- Panose-1 currently Latin only
- Currently best describe Latin, Greek, Cyrillic
Practical example: the XML

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="foo.css"?>
<help xml:lang="en">
    <title>Toy Help System</title>
    <topic class="boring">This topic has mixed content,
        however it is excruciatingly dull. Here is a
        procedure to follow.
        <procedure>
            <step>one</step>
            <step>two</step>
        </procedure>
    </topic>
    <topic class="funny">'A hazelnut in every bite' as
        the old advert used to say</topic>
</help>
```

Practical example: the CSS

```css
help, title, topic, subtopic, procedure
{ display: block; margin: 14px }
help
{ color: black; background: #DFD; font-size: 18px;
    margin: 0; font-family: Georgia, Times, serif;
    padding: 20px 10%; }
title
{ margin-bottom: 34px; background: #633; color: white;
    padding: 1em; border: thick solid #393;
    font: 28px/32px Arial, sans-serif }
.funny { background: #FFD}
topic{
    margin-left: 40px; border: thin solid silver;
    padding: 8px; text-align: justify }
subtopic { clear: left }
step { display: list-item; margin-left: 1em; color: green }
```
**Generated content and counters**

- **content** property generates content
- Pseudo-elements **:before** and **:after** style generated content
- Counters (with increment and reset), for numbering

```css
procedure {
    margin-left: 2em; padding: 8px; background: silver;
    border-right: 2px solid black;
    border-bottom: 2px solid black;
    border-top: 2px solid white;
    border-left: 2px solid white }

.funny:after { color: red; content: "!!!"}

.topic:before {
    content: "TOPIC " counter(top) ".";
    counter-increment: top; counter-reset: subtop;
    color: white; padding: 5px; background: #333;
    float: left; width: 90px; margin: 12px 30px 12px -40px }

.subtopic:before {
    content: "Subtopic " counter(top) "." counter(subtop) ": ";
    counter-increment: subtop; font-style: italic}
```
Practical example: result

Topic 1: This topic has mixed content, however it is excruciatingly dull. Here is a procedure to follow:

- one
- two

Topic 2: 'A hazelnut in every bite' as the old advert used to say!!!
**CSS Object Model**

- DOM Level 1 defined an Object Model for XML (and for HTML)
- DOM Level 2 defines
  - An event model, of which more later
  - An Object Model for CSS
    - Manipulate stylesheet(s)
    - Add stylesheets to a collection
    - Write to a 'run-time' stylesheet
    - Get cascaded value of any property

---

**Here Be Dragons**

The preceding slides have described stable W3C specifications

*Following slides describe ongoing work for CSS3, which can and likely will change! Exercise caution.*
CSS3: Paged media

(CSS2 already supports paged media such as printers and projectors)

- Headers and footers
- Footnotes and endnotes
- Duplex layout

```css
@media projection, print {
  H1 { page-break-before: always; margin-bottom: 0.8em}
}
```

CSS3: Generated content

(CSS2 already supports simple generated content before and after elements, including auto-generated counters)

- Date, document URL, page number are added
- Cross-references can be automatically inserted
- Generated content in margins
- Named strings (similar to counters)
CSS3: Multiple Columns

- Number of columns (greater than one!)
- Column widths and inter-column gaps
- Width, style and color of inter-column rules
- Column spanning

```css
story.article {
    column-number: 3;
    column-width: 10em 25% auto;
    column-rule: thin solid red
}
```
CSS3: Color Profiles

(CSS2 describes color in sRGB, an International Standard)

- Use other color profiles
- Specify/override color profile of images
- Specify rendering intent
- Named colors extended from 16 to 143

CSS3: International layout

(CSS2 already dealt with BIDI, and was Unicode-enabled)

- Layout grids for ideographic text
- More line breaking options (eg for Thai)
- Ruby formatting
- Improved text justification control
- Kashida for justifying cursive (eg, Arabic) text
**CSS3: User Interface**

(CSS2 already had some user-interface colors and fonts)

. Describe form elements using CSS
. Similar level of control to HTML form elements, but for XML
. Full-screen and kiosk-mode enhancements
. Make elements accept user input
. Media preferences

**CSS3: Behavioural Extensions**

. Standalone, declarative behavioural components
. CSS selectors attach behaviours to elements
. XML DTD need not be altered
. Different presentation forms can have different behaviours
. Promotes re-use of behaviour components

```javascript
@script "text/javascript" url("myscript.js");
header > TOCtitle {
  onclick : "fold_or_unfoldTOC(event)"
}```
**CSS3: Namespace support**

(CSS2 works with namespaced documents, but elements from all namespaces are matched)

- Matches on namespace URI
- Matching on namespace prefix is fragile, and not allowed

```css
@namespace foo url(http://www.foo.com);

foo|h1 { color: blue }
foo|* { color: yellow }
|h1 { color: red }
|h1 { color: green }
|hl { color: green }
foo\:hl { is: incorrect }
```
CSS and SVG

- SVG uses CSS properties, cascading, inheritance, selectors
- SVG uses its own rendering model, not a flow box model
- Horizontal and Vertical text supported
- New properties, for graphics:
  - `fill`, `stroke`
  - `opacity`, `pointer-events`, `filter`
- Aural and text-only rendering also possible

SVG: Hello World

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG February 2000//EN"
  "http://www.w3.org/Graphics/SVG/SVG-20000202.dtd">
<svg width="600" height="400">
  <defs>
    <style type="text/css">
      * {stroke: #336; }
      text { font-family:'Georgia', serif;
        fill:#39D; font-size:150}
    </style>
  </defs>
  <g>
    <rect x="30" y="30" width="540" height="340"
        style="fill:#F80"/>
    <text x="110" y="170">Hello</text>
    <text x="85" y="340">World</text>
  </g>
</svg>
```
SVG: CSS linking back to XML

- CSS2 defines a `cursor` property
  - format not constrained
  - SVG cursor element has hotspot, points to PNG
- SVG `fill` property can point to SVG gradient element
- SVG `filter` property points to SVG filter element
Fonts in SVG

- Fonts are basically Bezier curves and metadata
- SVG can render cubic and quadratic Bezier curves
- CSS2 lists the metadata required
- Adding `font`, `glyph` and `kern` elements was easy
- SVG implementations can help solve the font problem

SMIL Animation

- 'Dynamic HTML' is animation of HTML and CSS using DOM and script
- Editing tools can't read and understand a script
- Declarative animation is portable and efficient
- SMIL animation for XML attributes and CSS properties
- Built on XML DOM and CSS OM, but no script

```xml
<rect ...>
  <set begin="onclick" dur="7s" to="red"
      attributeName="fill" attributeType="CSS" />
</rect>
```
**Summary: CSS supports**

- Progressive rendering
- Dynamic modification (script)
- Declarative animation (SMIL)
- Downloadable fonts
- Stylable graphics
- Visual and aural rendering
- Cascading reader/author balance

*CSS is specifically designed for the Web*

---

**CSS doesn't do:**

- Complex queries
- Rearranging elements
- Combining documents
- Creating TOCs
- Manipulating documents
- Report generation

*DOM gives access to CSS properties and XML*

*XSL-T can transform between XML grammars*
Testing CSS

- CSS1 test suite
  - http://www.w3.org/Style/CSS/Test
- CSS2 test suite is being developed
- SVG test suite
  - http://www.w3.org/Graphics/SVG/Test
- CSS Validator
  - http://jigsaw.w3.org/css-validator/

For more information

CSS overview
  - http://www.w3.org/Style/CSS
CSS2 Recommendation
  - http://www.w3.org/TR/REC-CSS2
CSS3 tracking
  - http://www.w3.org/Style/CSS/current-work
XSL Overview
  - http://www.w3.org/Style/XSL
SVG Overview
  - http://www.w3.org/Graphics/SVG