Separation of Style and Content

- CSS = Cascading Style Sheets
  - – cumulative effect with overrides, hence "cascading"
  - – "A simple mechanism for adding style to Web documents"
    - fonts, colours, spacing, …
- CSS applies generically to all forms of XML
- Particularly optimized for use with HTML
  - – In original HTML, style was done using HTML elements and attributes.
  - – In HTML 4, these were mostly deprecated in favour of attaching to *external style languages* not defined as part of HTML.
    - CSS is an example
    - Definitive spec:
      - – Cascading Style Sheets, level 1
        - » http://www.w3.org/TR/REC-CSS1

CSS Levels

- CSS Level 1 (implemented in NN4/6 and IE4/5)
  - – Dec, 1996
  - – simple
- CSS Level 2 (bits and pieces being implemented)
  - – May 1998
  - – more selectors, media-specific style sheets, content positioning, downloadable fonts, table layout, …
- CSS Level 3 (nowhere near being implemented)
  - – still under development
  - – modularization
  - – form elements, full-screen, fine-control of color, multi-column layout, paged media,

Linking CSS to XML

- Use the `<!xml-stylesheet` PI
- Add "style" attributes
  - but must declare in DTD
- resume.xml
  - resume.css
  - resume.dtd

Linking HTML with style sheets

```
<html>
  <head>
      <title>title</title>
      <link rel="stylesheet" type="text/css"
            href="http://style.com/cool"/>
      <style type="text/css">
          @import url(http://style.com/basic);
          h1 { color: blue }
      </style>
  </head>
  <body>
    <h1>Headline is blue</h1>
    <p style="color: green">While the paragraph is green.</p>
  </body>
</html>
```

The CSS Language

- Using CSS, one associates style property values with XML elements.
  - by element kind
    - <p>, <h1>, …
  - by element "class"
    - <p class="my-arbitrary-class-name">
  - by individual element
    - <p id="a-particular-paragraph">
  - by element in context
    - <h1>This <em>word</em> is emphasized

CSS Grammar

```
stylesheet: import* ruleset*

import: 'import' 'url(' url ')'

ruleset:
   selector [ ',' selector]*
        '{' declaration [ ';' declaration]* '}'

selector: simple_selector+

declaration: property ':' expr ['! important']?

expr: ['+' | '-']? term [ ['/' | ',' | ' ' ] term ]*
```

Grouping

- One can group multiple headings, and apply a set of styles to all.
  - h123.html
  - h123css.html

Inheritance

- Most styles are inherited into nested elements.
- Can set a "default" style by setting property values for the <body> element.
  - greencss.html

Class as Selector

- HTML elements can be tagged with possibly many classes.
- Style properties can bet set across all elements of a given class.
  - classcss.html

ID as Selector

- HTML4 and XML allows any element to have a special "id" attribute that is unique in the document.
  - Can be used as the target for a hyperlink
  - Can be used to associate style properties with a particular element
    - idcss.html

Contextual Selectors

- CSS can match a search pattern on a stack of open elements.
  - contextcss.html
- Ancestors, not just parents.
- Can mix and match the various types of selectors into sentences:
  - #x23a p .foo { color: red; }

Anchor Pseudo-Classes

- A hack to account for elements whose behaviour changes through time.

```
a:link { color: red } /* unvisited link */
a:visited { color: red } /* visited link */
a:active { color: red } /* active link */
```

  - No effect on elements other than <a>, so 'a' may be omitted.

Typographical Pseudo-Elements

- Another hack for typographically important regions of text that are not delimited.
  - First-line formatting

    ```
    P:first-line { font-variant: small-caps }
    ```

  - First-letter (drop-caps) formatting

    ```
    P:first-letter { font-size: 200%; float: left }
    ```

    - browser defines what's "in" the first letter (e.g., opening quotes).
- "Pseudo" because it's as if the browser inserted these elements (and classes) for us.
  - pseudocss.html

The Cascade

- >1 style sheet can influence the presentation simultaneously.
  - modularity
  - author/reader balance
- For different properties, all matching selectors are applied
- In the case of a property conflict
  - each rule is assigned a weight
  - heaviest wins

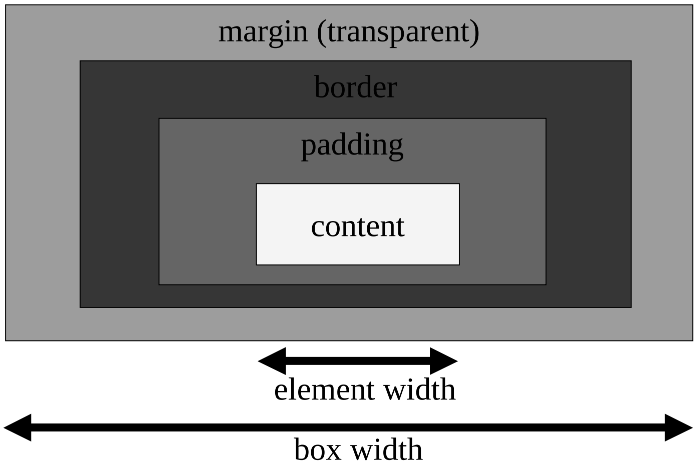
Weight Algorithm

1. Find all declarations that apply to the element/property
   - if none, then inherit
     - if no inheritance, use initial value
2. Sort by presence of 'important'
3. Sort by origin (author or reader)
   - author overrides reader overrides browser default
4. Sort by specificity of selector
   - more specific wins ('h1 p' wins over 'p')
   - in-element ('style' attribute) is the same specificity as an id-based selector
5. Sort by order specified
   - latter specified wins

Classification properties

- CSS works entirely internally to classify elements.
  - – relies only on default values of certain properties
  - – can make a <p> into a <li> (for formatting purposes) if you want.
- They are:
  - – display: block | inline | list-item | none
    - none: turns off display
    - block: opens a new box (<div>) (new-line before and after)
    - list-item: same as block with a list-item marker
    - inline: box that can be broken at line-breaks (<span>)
  - – white-space: normal | pre | nowrap
  - – list-style-type, list-style-image, list-style-position, list-style

Box Formatting Model



margin (transparent)

border

padding

content

element width

box width

Box Properties

- margin-top, margin-bottom, margin-left, margin-right, margin
- padding-top, padding-bottom, padding-left, padding-right, padding
- border-top-width, border-bottom-width, border-right-width, border-left-width

Positioning

- Vertical
  - margin width specifies min. distance to the edges of surrounding boxes
    - uses max(left margin, right margin)
    - verticalcss.html
- Horizontal
  - not collapsed as for vertical
  - various "auto" values used to make sum of all the width elements equal to the enclosing parent width
  - horizontalcss.html

Floating Elements

- float: left | right | none
  - left, right: formatted as block
    - left: moved to the left, text wraps on the right
    - right: moved to the right, text wraps on the left
- clear: none | left | right | both
  - list sides where floating elements are not accepted
    - e.g., if 'left'. element will be moved below any left floating element
- floatcss.html

Font Properties

- The UA (user agent = browser)
  - makes a list of properties of all known fonts (system installed or previously downloaded)
  - for each element, UA assembles font-properties applicable to the characters of that element
    - uses the 'font-family' to choose a tentative font
    - remaining properties are tested against it (e.g., size, italic, …)
    - ensure glyph available for current character
    - if matches, use it
  - if no match, repeat with next alternative matching font family
  - if no matches at all, use a UA-dependent default font-family

Font Matching

- font-style: normal | italic | oblique
  - italic matches if keyword italic or oblique found
  - else must match exactly
- font-variant: normal | small-caps
  - small-caps satisfied if keyword present or can be synthesized
- font-weight: normal | bold | bolder | lighter | 100-900
  - always matches
- font-size: *absolute* | *relative* | *length* | *percentage*
  - matches within UA-defined tolerance
    - scalable fonts matched to within pixel
    - bitmapped fonts matched within as much as 20%

Font Families

- font-family: [family-name | generic-family] [, [family-name | generic-family]]*
  - e.g.,
    - body { font-family: gill, helvetica, sans-serif }
  - generic-families:
    - serif
    - sans-serif
    - cursive
    - fantasy
    - monospace
- `fontcss.html`

Text Properties

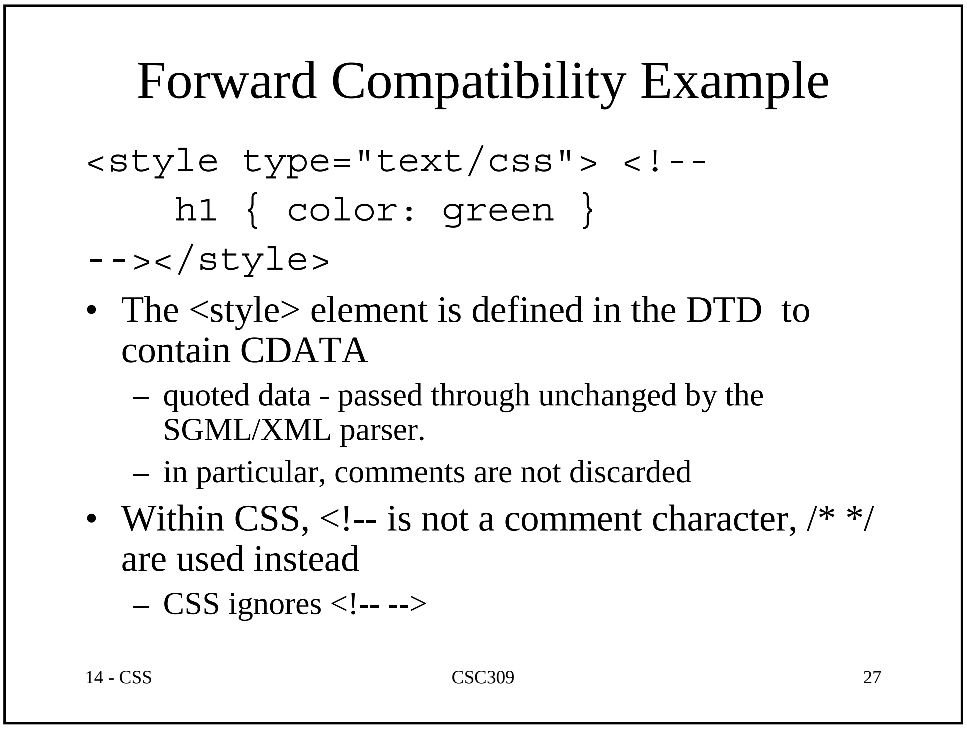
- properties:
  - word-spacing: normal | *length*
  - letter-spacing: normal | *length*
  - text-decoration: underline | overline | line-through | blink
  - text-transform: capitalize | uppercase | lowercase | none
  - text-align: left | right | center | justify
  - text-indent: *length* | *percentage*
  - line-height: *number* | *length* | *percentage*
- textcss.html

Vertical Alignment

- vertical-align: baseline | sub | super | top | text-top | middle | bottom | text-bottom | *percentage*
- textvaligncss.html

Forward Compatibility

- Older browsers will ignore tags they don't recognize.
  - However, they will treat content as text to be rendered.
- Need to enclose content in SGML comment delimiters
  - <!-- … -->
- Problem: won't new browsers also treat it as a comment?



Forward Compatibility Example

```
<style type="text/css"> <!--
    h1 { color: green }
--></style>
```

- The <style> element is defined in the DTD to contain CDATA
  - quoted data - passed through unchanged by the SGML/XML parser.
  - in particular, comments are not discarded
- Within CSS, <!-- is not a comment character, /* */ are used instead
  - CSS ignores <!-- -->