

Cascading Style Sheets

CMPS 183 - Spring 2003
Hypermedia and the Web

Reference: "Cascading Style Sheets - The Definitive Guide" by Eric Meyer

Why CSS?

- HTML was originally designed as a structural markup language
 - The display style for each tag was not specified by the language.
- Mosaic, then Netscape and IE added display style tags, like ``.
 - These new tags do not impart any structural knowledge.
 - No way to tell between `<H1>` and ``.

What is CSS?

- Cascading Style Sheets
 - Cascading: The most specific scope takes precedence.
 - Style: Specifies the formatting and style for HTML structure tags
 - Sheets: The styles can be saved in a common file and referenced by all pages in a site. They can also be embedded in a single web page.

An Example

- `H1 {color: maroon; font: italic 1em Times, serif; text-decoration: underline; background: yellow;}`
 - text color = maroon
 - font = italicized Times, serif
 - style = underlined
 - background color = yellow

A Bigger Example

- `H1 {color: maroon; font: italic 1em Times, serif; text-decoration: underline; background: yellow url(titlebg.png) repeat-x; border: 1px solid red; margin-bottom: 0; padding: 5px;}`
 - Adds a background image that repeats horizontally
 - 1 pixel red border around the text (minimum 5 pixels offset)
 - No margin at the bottom of the tag (most browsers put a blank line after an H1 tag).

Referencing a Style Sheet in HTML

- Inside the `<HEAD></HEAD>` tags
- `<LINK REL="stylesheet" TYPE="text/css" HREF="sheet1.css" TITLE="Default">`
 - REL: stands for relation; "stylesheet" is normal, "alternate" allows other style options.
 - TYPE: always "text/css"
 - HREF: just like the `<A>` tag
 - TITLE: names the style; some browsers allow individual styles to be enabled and disabled.

Another way to Reference

```
<STYLE TYPE="text/css"><!--  
  @import url(sheet2.css);/*Start with the default*/  
  H1 { color: maroon; }  
  BODY { background: yellow; }  
--></STYLE>
```

- `<STYLE></STYLE>` allows specification of page specific styles, the TYPE attribute is required, with the "text/css" value
- @import: load external style sheet(s)
- list individual styles after
- HTML Comment tags help older browsers

Inline Styles

- `<P STYLE="color:maroon; background:yellow;">`
 - Sets the style for just that paragraph.
 - You can use this method to set the style of *any* HTML element found inside the `<BODY></BODY>` tags.

Rules

- H1 { color: blue; }
- General case: [SELECTOR] { [PROPERTY]: [VALUE]; }
- Obvious selectors are from HTML. CSS can also be applied to XML files where selectors can be anything.
- Values can be a single word, or space separated words applicable to that property. Only a few properties allow this (the `font` property for example).
- The semicolon is required to terminate the statement. Do not forget the last one when listing multiple pairs.

Grouping Selectors

- List multiple selectors (comma separated) before the property list to apply the properties to multiple tags.

- `H1, H2, H3, H4, H5, H6 { color: purple; }`

```
H1 { color: purple; background: yellow; }
```

```
H2 { color: purple; background: white; }
```

```
H3 { color: blue; background: white; }
```

```
H4 { color: blue; background: yellow }
```

- Or

```
H1, H2 { color: purple; }
```

```
H3, H4 { color: blue; }
```

```
H1, H4 { background: yellow; }
```

```
H2, H3 { background: white; }
```

Styling Classes

- So far, the styles have applied to a specific element.
- Sometimes, you want to create styles that are element independent, and can be applied to multiple (but not all) instances of an element (or elements).

```
<P CLASS="warning">Pop Quiz Today!</P>  
<P>No really, there is a <SPAN CLASS="warning">Pop  
Quiz Today</SPAN>!</P>
```

```
.warning {font-weight: bold;}
```

- You can also apply these to just one type of element:

```
P.warning {font-weight: bold;}
```

Multiple classes

- You can combine classes together:

```
.warning {font-weight: italic;}  
P.warning {font-weight: bold;}
```

- All warnings will be in italics, but only those in a P element will also be bold.

ID Selectors

- You can also use the ID attribute and assign styles based on the value of the ID.

```
#first-para {font-wieght: bold;}
```

```
<P ID="first-para">blah blah blah</P>
```

```
<P>second blah blah blah</P>
```

- Note that ID selectors are preceded by a # sign instead of the . sign.
- Only the first P will be bold.

Classes... IDs...

HELP!!!!

- In HTML, IDs are supposed to be unique in a document, while classes can be in many places.
- IDs have a higher weight than Classes

Pseudo-classes

- Pseudo-classes allow you to style elements based on their context in the page, such as a non-visited or visited link.
- CSS1 defines three pseudo-classes:
 - `:link` - an unvisited hyperlink
 - `:visited` - a visited hyperlink
 - `:active` - a hyperlink in the process of being activated. In CSS2, this can be applied to any element (what active means for a paragraph might open to interpretation).

Using Pseudo-classes

```
A:link { color: blue; }  
A:visited { color: red; }  
A:active { color: yellow; }
```

- The :link differs from plain A styles because it applies only to hyperlinks, whereas the plain A style applies also to named anchors.
- You can use pseudo-classes with real classes:

```
A.external:link { color: green; }
```

```
<A CLASS="external" HREF="http://www.sun.com">Sun</A>
```

- Remember not all browsers can handle active redisplay

Pseudo-elements

- You can apply a different style to the first letter and/or the first line of an element with pseudo-elements
- `:first-letter` - a visited hyperlink
- `:first-line` - a visited hyperlink
- Not all properties can be applied to pseudo-elements, but all font, color, and background properties are allowed.

Contextual selectors

- You use contextual selectors when you want to apply a style based on the containment of the elements.
- For example, you can make `` to mean bold in a paragraph, and italics in a heading:

```
H1 STRONG { font-weight: italic; }  
P STRONG { font-weight: bold; }
```

- You can put as many levels into the contextual selector as you want.
- You can also specify classes for each element in the list.

Inheritance

- Most, but not all, properties are inherited based on containment.
- `<H1>This is a big deal.</H1>`
- Properties applied to the `<H1>` tag that aren't overridden by properties on the `` tag will be applied to all the text inside the `<H1></H1>` tags.
- Borders are not inherited.

Conflict resolution

```
.purple { color: purple; }  
H1 { color: green; }
```

```
<H1 CLASS="purple">What color am I?</H1>
```

- CSS defines numeric specificities for each type of rule:
 - Simple selectors = 1
 - Class selectors = 10
 - ID selectors = 100
- In the example here, the text is purple.
- P.bright EM.dark {color: brown;} has a specificity of 22.

Important rules

- Sometimes you want a rule to override everything else, no matter what the specificity of any other rules that might apply is.

```
STRONG { color: red !important; font-weight: bold; }
```

- Here, the color of any text in a STRONG tag will always be red, but the font-weight can be overridden by other rules.
- If all the properties were important, each one would need a !important flag.

An Important Exception

- When the important property is inherited, it can be overridden by properties of the inner tag:

```
H1 { color: blue !important; }  
EM { color: red; }
```

```
<H1>Here the text is blue <EM>and here its red.</  
EM></H1>
```

- Inherited values always have a specificity of 0.

When Rules Collide

```
H1 <color:blue;>
```

```
H1 <color:red;>
```

- How the winner is picked:
 1. Find all declarations that contain a selector that matches the element
 2. Sort by explicit weight, with !important rules being placed higher. Also sort by origin: author, reader, and user agent (in most cases, the author wins, and user agent always loses).
 3. Sort by specificity. Elements with a higher specificity have a higher weight.
 4. Sort by order of appearance. More recent rules win.

Element types

- Block level elements: paragraphs, headings, lists, tables, DIVs, and BODY.
 - These elements can only be children of other block level elements, and not under all circumstances.
- Inline elements: A, EM, SPAN, and most replaced elements (like images and input forms).
 - These elements do not force new lines and can be children of any other element.
- List item elements: Only LI

The Display Property

- You can use the display property to alter how things are shown in the page.
- There are four options: block, inline, list-item, none
- By setting the style on a <P> tag to “display:inline;”, you can prevent any white space; <P> would appear to the user in the same way a tag does.
- Setting the style on an tag to “display:block;” will force images to be on a new line.
- Setting a display style to none will hide the element from the user.

More Display

- You can use this tag to totally disrupt the standard structural flow of an HTML document.
 - DON'T! Most browsers won't handle it, and might crash, or just look really really bad.
- When using alternate style sheets (which many browsers don't support well, if at all), you can hide some information in one style, and display it in another (think of "short" style versus "verbose" style).
- In CSS2, table related values were added to the display tag.

Specifying color

- There's three major ways to specify a color:
 - Hexidecimal (like normal HTML) - #FFFF00
 - Percentage - rgb(100%, 100%, 0%)
 - Numeric - rgb(255, 255, 0)

Absolut! Units

- Insert imaginative vodka art here...
- Inches - in
- Centimeters - cm
- Millimeters - mm
- Points - pt
 - Standard typographic measure: 72 pt / 1 in
- Picas - pc
 - Also from typography: 1 pc = 12 pt, 6 pc / 1 in

Relative units

- em-height - em
 - Can be used to set relative font sizes `{font-size:0.8em;}`
 - Also refers to the margin
- ex - x-height
 - Literally - this refers to the size of the lowercase letter
x
- px - pixels
- Practically, em and px are the most useful units

Percentages and URLs

- Percentages are obvious
 - A positive number, followed by %
- URLs
 - Absolute - `url(http://www.sun.com/style.css)`
 - Relative - `url(mystyle.css)`
 - Not all browsers correctly handle relative urls.
 - There can be no spaces between “url” and the opening parenthesis

CSS2 Units

- Angle units: rad (radians), deg (degrees), grad (grads)
- Time units: s (seconds), ms (milliseconds)
- Frequency units: Hz (hertz), mHz or mhz (megahertz)

Text Options

- `text-indent: [length] | [percentage]`
 - Negative values are allowed
- `padding-left: [length]`
 - Forces padding on the left of the text
- `text-align: left | center | right | justify`
 - To center an image with `text-align`, you must wrap a DIV around it, and apply the property to the DIV.
 - `<DIV STYLE="text-align:center;"></DIV>`

Text Options

- `white-space: pre | nowrap | normal`
 - `pre` - Pre-formatted, spaces are displayed as typed
 - `nowrap` - Only the `
` tag can cause a new line
- `line-height: [length] | [percentage] | [number] | normal`
 - lengths and percentages can be problematic when they are inherited from a smaller font to a bigger one
 - Using a plain number acts like a scale factor and is a better choice.

Text Options

- `vertical-align: baseline | sub | super | bottom | text-bottom | middle | top | text-top | [percentage]`
 - Good for aligning text with images.
 - Doesn't apply to table cells in CSS1
 - baseline is normal
 - sub = subscript, super = superscript
 - bottom and top goes to the extremes of the line, possibly below or above the text

Text Options

- `word-spacing: [length] | normal`
 - Adds some amount of space between words
- `letter-spacing: [length] | normal`
 - Adds some amount of space between letters
- `text-transform: uppercase | lowercase | capitalize | none`
 - `capitalize` makes the first letter of each word capitalized

Text Options

- text-decoration: none | [underline || overline || line-through || blink]
 - Everything but “none” can be combined
 - Separate with a space
 - Browsers may or may not allow the decorations to be inherited. For example, if you underline a paragraph, but have text inside with no decoration, there is no way know if users will see an underline on the text or not.

Fonts

- There is no way to be sure a font you specify is available on everyone's computer, or that it will have the same name as it does on yours (Times New Roman, TimesNR, Times).
- CSS allows you to specify a font family (like Times) to refer to a number of actual fonts (TimesBold, TimesItalic, TimesRegular, etc.)

Fonts

- CSS defines generic font families:
 - serif fonts: proportional, with serif decorations
 - sans serif fonts: proportional without serifs
 - monospace fonts: not proportional, like a typewriter
 - cursive fonts: tries to look like human handwriting
 - fantasy fonts: unclassifiable, like Western, Klingon

Fonts

- `font-family: [family name | generic-family]*`
 - Allows order of preference specification, the first one that matches a known font on the user's system is used
 - `H1 { font-family: Garamond, 'New Century Schoolbook', serif; }`
 - Quotes are needed if the name includes spaces or symbols (`%`, `#`, `$`, etc.)
 - Only one generic family can be specified at the end
 - Many systems don't have any cursive or fantasy fonts

Fonts

- `font-weight: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900`
 - The numbers are not reliable, they only say that a given weight will be no lighter than the one preceding it.
 - 400 = normal, 700 = bold, 500 = medium (when available)
 - 300 will be lighter than 400, if a lighter font is available (same for 200 and 100)
 - Try out the weights in a real web page to see

Fonts

- `font-size: xx-small | x-small | small | medium | large | x-large | xx-large | smaller | larger | [length] | [percentage]`
 - medium is default
 - Be careful using length values here because different systems compute the lengths differently, most can't determine how many pixels per inch are actually being displayed.
 - smaller and larger will be applied to inherited values
 - percentage will be calculated and rounded as appropriate

Fonts

- `font-style: italic | oblique | normal`
 - italic text is a little different from the normal text to account for the slant
 - oblique text is the normal text, just slanted
 - In some fonts, there's no difference between them
- `font-variant: small-caps | normal`
 - small-caps uses all capital letters, but uses larger font sizes for letters that are capitalized
 - I Am Myself = I AM MYSELF

Fonts

- `font: [font-style || font-variant || font-wieght]? font-size [/ line-height]? font-family`
 - This is shorthand for the combination of properties
 - `H2 {font: bold normal italic 24px Veranda, Arial, sans-serif; }`
 - The first three, if used, can be in any order
 - font-size and font-family are required, in order
 - line-height requires a / between the size and height
 - `H2 { font: 24px/1.2 sans-serif; }`

Colors

- Foreground colors usually apply to text and borders
- `border-color: [color];`
 - Overrides the foreground color for the border
- Sometimes colors can be applied to forms
 - CSS2 allows different colors for different form objects
- `background-color: [color] | transparent`
 - If you set a different background color, consider using the `padding` property to create some blank space

Backgrounds

- `background-image: [URL] | none`
 - The actual URL goes in parens
 - ```
P {background-image: url(http://www.someplace.com/images/light_background.jpg); background-color: yellow; color: black;}
```
  - Relative URLs should be interpreted based on the stylesheet's location; not all browsers do this correctly
  - Use a background color with a background image in case the image isn't found or available (saved pages)

# Backgrounds

- `background-repeat: repeat | repeat-x | repeat-y | no-repeat`
  - repeat is the default
- `background-position: ([percentage] | [length]){1,2} | [ top | center | bottom ] || [ left | center | right ]`
  - If you only use one keyword, the other is assumed to be center: top = top center, left = left center
  - percentages are based on the location parent
  - You can mix percentage and length, but not with percentage and keywords or length and keywords.

# Backgrounds

- background-attachment: scroll | fixed
  - fixed - Allows you to keep the background on screen, even if the user scrolls the text
  - scroll - The default value, allowing the image to scroll off screen
- background: [background-color] || [background-image] || [background-repeat] || [background-attachment] || [background-position]
  - If you use both background-position values, they must appear together, horizontal first, then vertical

# Boxes and Borders

- All elements have a height and a width.
  - `width: [length] | [percentage] | auto`
  - `height: [length] | auto`
  - Heights are automatically calculated, widths are no wider than the parent
- Outside the area determined by height and width there are a number of pre-defined areas where you can add spacing or set parameters
  - From closest to farthest: inner edge, padding, border, margin, outer edge

# Boxes and Borders

- Background images extend into the padding, but not the margin
- `margin: ([length] | [percentage] | auto){1,4}`
  - You either specify one margin for all sides, or four margin values for top right bottom left (space separated, in that order)
  - If you don't enter a value for bottom, the top value is used; if you don't enter a value for left, the right value is used.
- `margin-top / margin-right / margin-bottom / margin-left`

# Boxes and Borders

- Margins will be collapsed together (the largest margin between two spaces will be used)
  - padding and borders are not collapsed
- Margins can take negative values
- There are lots of browser bugs related to margins; you have been warned.

# Boxes and Borders

- `border-style:` (none | dotted | dashed | solid | double | groove | ridge | inset | outset) {1, 4}
- You can give different styles for different sides: top  
right bottom left
- unlisted values work the same as they did for margins
- Browsers are allowed to display any border other than none as a solid

# Boxes and Borders

- `border-width: (thin | medium | thick | [length])`  
`{1, 4}`
  - Again you can give one or more widths.
- `border-top-width / border-right-width / border-bottom-width / border-left-width`
- When you set the border to none, the border-width is ignored
- `border-color: [color] {1, 4}`
  - Again you can give one or more colors

# Boxes and Borders

- `border-top: [border-width] || [border-style] || [color]`
  - `border-right`, `border-bottom`, `border-left`
  - `border` (the simplest shorthand)
    - Apply `border` first, then override with details

## Boxes and Borders

- `padding: ([length] | [percentage]) {1, 4}`
  - Again, one or more values
- `padding-top / padding-left / padding-bottom / padding-right`

# Floating and Clearing

- `float: left | right | none`
  - Images are most commonly floated, but CSS1 allows any element to be floated (headings, paragraphs, etc.)
- `clear: left | right | both | none`
  - Makes sure a floated element is cleared before the next one begins

# Lists

- `list-style-type: disc | circle | square | decimal | upper-alpha | lower-alpha | upper-roman | lower-roman | none`
- `list-style-image: [URL] | none`
  - Uses an image for each bullet
  - Again, use a style image with a style type in case the image isn't available
- `list-style-position: inside | outside`
  - Allows you to put the bullet inside the text area (where the wrapping text will be under it)