CSS

Applications to HTML

The problem with HTML

- HTML was originally intended to describe the *content* of a document
- Page authors didn't have to describe the layout--the browser would take care of that
- This is a good engineering approach, but it didn't satisfy advertisers and "artists"
  - Even people that actually had something to say wanted more control over the appearance of their web pages
- As a result, HTML acquired more and more tags to control *appearance*
  - Content and appearance became more intertwined
  - Different browsers displayed things differently, which is a real problem when appearance is important

Cascading Style Sheets

- A Cascading Style Sheet (CSS) describes the appearance of an HTML page in a separate document
- CSS has the following advantages:
  - It lets you separate content from presentation
  - It lets you define the appearance and layout of all the pages in your web site in a single place
  - It can be used for both HTML and XML pages
- CSS has the following disadvantage:
  - Most browsers don't support it very well

CSS syntax, I

- CSS syntax is very simple--it's just a file containing a list of selectors (to choose tags) and descriptors (to tell what to do with them):
  - Example: `h1 {color: green; font-family: Verdana}` says that everything included in `h1` (HTML heading level 1) tags should be in the Verdana font and colored green
- A CSS file is just a list of these selector/descriptor pairs
  - Selectors may be simple HTML tags or XML tags, but CSS also defines some ways to combine tags
  - Descriptors are defined in CSS itself, and there is quite a long list of them

CSS syntax

- The general syntax is:

  selector { ***property***: ***value*** }

  - or

    selector, . . . , ***selector*** {
    property: ***value***;
    . . .
    property: ***value***
    }

  - where
    - *selector* is the tag to be affected (the selector is case-sensitive if and only if the document language is case-sensitive)
    - *property* and *value* describe the appearance of that tag
    - Spaces after colons and semicolons are optional
    - A semicolon must be used *between* property:value pairs, but a semicolon after the last pair is optional

Example of CSS

```
/* This is a comment */
h1,h2,h3 {font-family: Arial, sans-serif;}   /* use 1st available font */
p, table, li, address {                      /* apply to all these tags */
  font-family: "Courier New";                /* quote values containing spaces */
  margin-left: 15pt;                         /* specify indentation */
}
p, li, th, td {font-size: 80%;}              /* 80% of size in containing element */
th {background-color:#FAEBD7}  /* colors can be specified in hex */
body { background-color: #ffffff;}
h1,h2,h3,hr {color:saddlebrown;} /* adds to what we said before */
a:link    {color:darkred}                    /* an unvisited link */
a:visited {color:darkred}                    /* a link that has been visited */
a:active  {color:red}                        /* a link now being visited */
a:hover   {color:red}                        /* when the mouse hovers over it */
```

Adapted from: http://www.w3schools.com/css/demo_default.htm

More about selectors, I

- As we have seen, an XML or HTML tag can be used as a simple element selector:

```
body { background-color: #ffffff }
```

- You can use multiple selectors:

```
em, i {color: red}
```

  You can repeat selectors:

```
h1, h2, h3 {font-family: Verdana; color: red}
h1, h3 {font-weight: bold; color: pink}
```

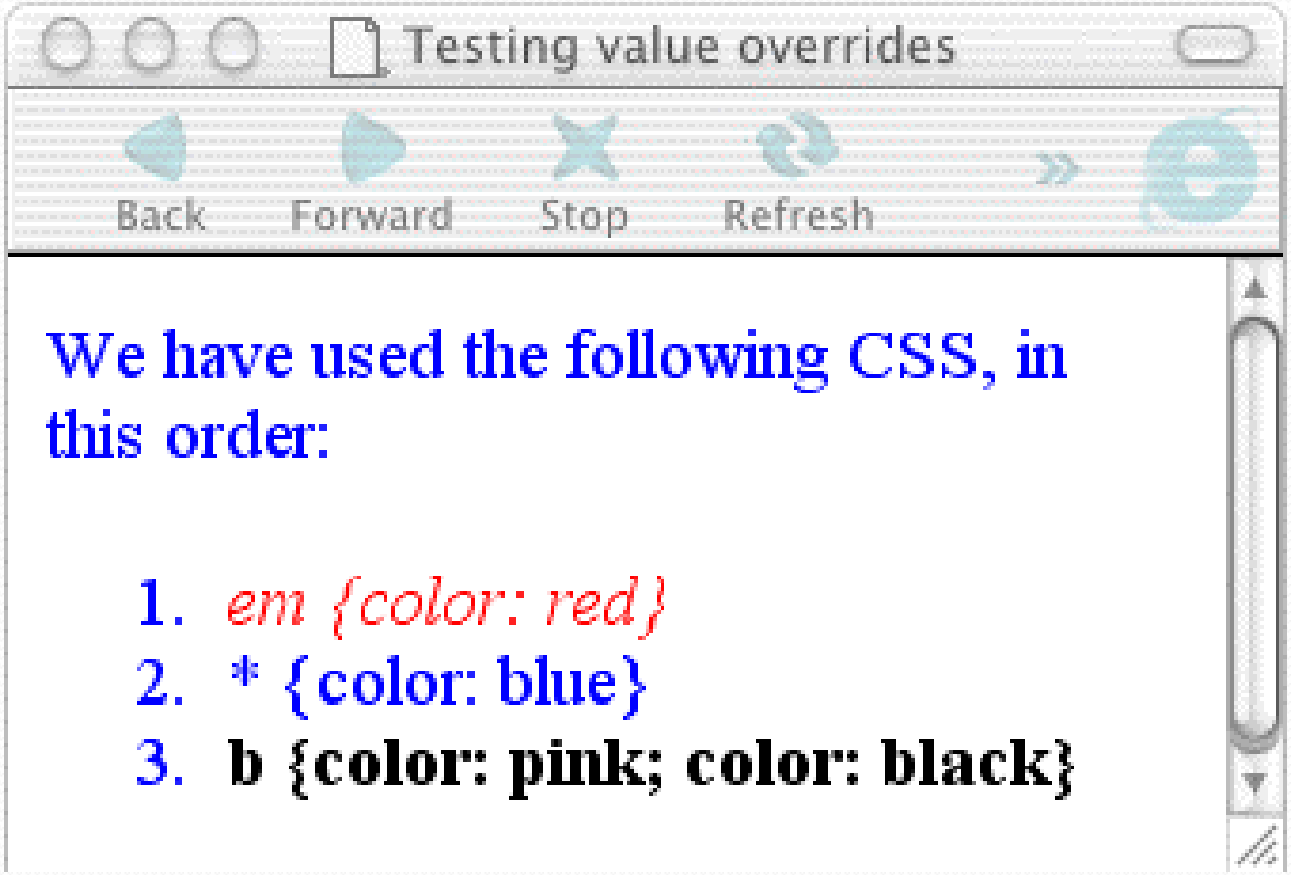
  - When values disagree, the last one overrides any earlier ones

- The universal selector * applies to any and all elements:

```
* {color: blue}
```

  - When values disagree, more specific selectors override general ones (so em elements would still be red)

Example of overriding



More about selectors, II

A descendent selector chooses a tag with a specific ancestor:

- p code { color: brown }
- selects a code if it is somewhere inside a paragraph

- A child selector > chooses a tag with a specific parent:

  h3 > em { font-weight: bold }

  selects an em only if its immediate parent is h3

- An adjacent selector chooses an element that immediately follows another:

  b + i { font-size: 8pt }

  Example: I'm bold and I'm italic

  Result will look something like: **I'm bold and** *I'm italic*

More about selectors, III

- A simple attribute selector allows you to choose elements that have a given attribute, regardless of its value:
  - Syntax: *element*[*attribute*] { … }
  - Example: table[border] { … }
- An attribute value selector allows you to choose elements that have a given attribute with a given value:
  - Syntax: *element*[*attribute*="*value*"] { … }
  - Example: table[border="0"] { … }

More about values

- As we have seen, the syntax for a CSS rule is:
  ***selector*, …, *selector* { *property*: *value*; . . . *property*: *value* }**
- The value is whatever occurs between the colon and the semicolon (or closing brace)
- Example: * {font-family: Trebuchet, Verdana, sans-serif;}
  - This means to use the Trebuchet font for everything, if it is available; else use the Verdana font, if available; else use whatever sans serif font the browser uses as default
- section {border: thin solid blue;}
  - This means to put a borders around section elements; the borders are to be thin *and* solid *and* blue

The class attribute

- The class attribute allows you to have different styles for the same element
  - In the style sheet:

```
p.important {font-size: 24pt; color: red}
p.fineprint {font-size: 8pt}
```

  - In the HTML:

```
<p class="important">The end is nigh!</p>
<p class="fineprint">Offer ends 1/1/97.</p>
```

- To define a selector that applies to any element with that class, just omit the tag name (but keep the dot):

```
.fineprint {font-size: 8pt}
```

The id attribute

- The id attribute is defined like the class attribute, but uses # instead of .
  - In the style sheet:
    p#important {font-style: italic} or
    # important {font-style: italic}
  - In the HTML:
    <p id="important">
- class and id can both be used, and do not need to have different names:
  <p class="important" id="important">

div and span

- div and span are HTML elements whose only purpose is to hold CSS information
- div ensures there is a line break before and after (so it's like a paragraph); span does not
- Example:
  - CSS: div {background-color: #66FFFF}
    span.color {color: red}
  - HTML: This div is treated like a paragraph, but this span is not.

Using style sheets

- There are three ways of using CSS:
  - External style sheet
    - This is the most powerful
    - Applies to both HTML and XML
    - All of CSS can be used
  - Embedded style sheet
    - Applies to HTML, *not* to XML
    - All of CSS can be used
  - Inline styles
    - Applies to HTML, not to XML
    - Limited form of CSS syntax

External style sheets

- In HTML, within the <head> element:
  <link REL="STYLESHEET" TYPE="text/css" HREF="***Style Sheet URL***">

- As a PI in the prologue of an XML document:
  <?xml-stylesheet href="***Style Sheet URL***" type="text/css"?>

- Note: "text/css" is the MIME type

Embedded style sheets

- In HTML, within the <head> element:

```
<style TYPE="text/css">
    <!--
    CSS Style Sheet
    -->
</style>
```

- Note: Embedding the style sheet within a comment is a sneaky way of hiding it from older browsers that don't understand CSS

Inline style sheets

- The STYLE attribute can be added to any HTML element:

  <*html-tag* STYLE="*property*: *value*"> or

  <*html-tag* STYLE="*property*: *value*;
  property: *value*; ...; *property*: *value*">

- Advantage:
  - Useful if you only want a small amount of markup
- Disadvantages:
  - Mixes display information into HTML
  - Clutters up HTML code
  - Can't use full range of CSS features

Cascading order

- Styles will be applied to HTML in the following order:
  1. Browser default
  2. External style sheet
  3. Internal style sheet (inside the <head> tag)
  4. Inline style (inside other elements, outermost first)
- When styles conflict, the "nearest" (most recently applied) style wins

Example of cascading order

- External style sheet:

```
h3 {    color: red;
        text-align: left;
        font-size: 8pt
}
```

- Internal style sheet:

```
h3 {    text-align: right;
        font-size: 20pt
}
```

- Resultant attributes:

```
color: red;
text-align: right;
font-size: 20pt
```

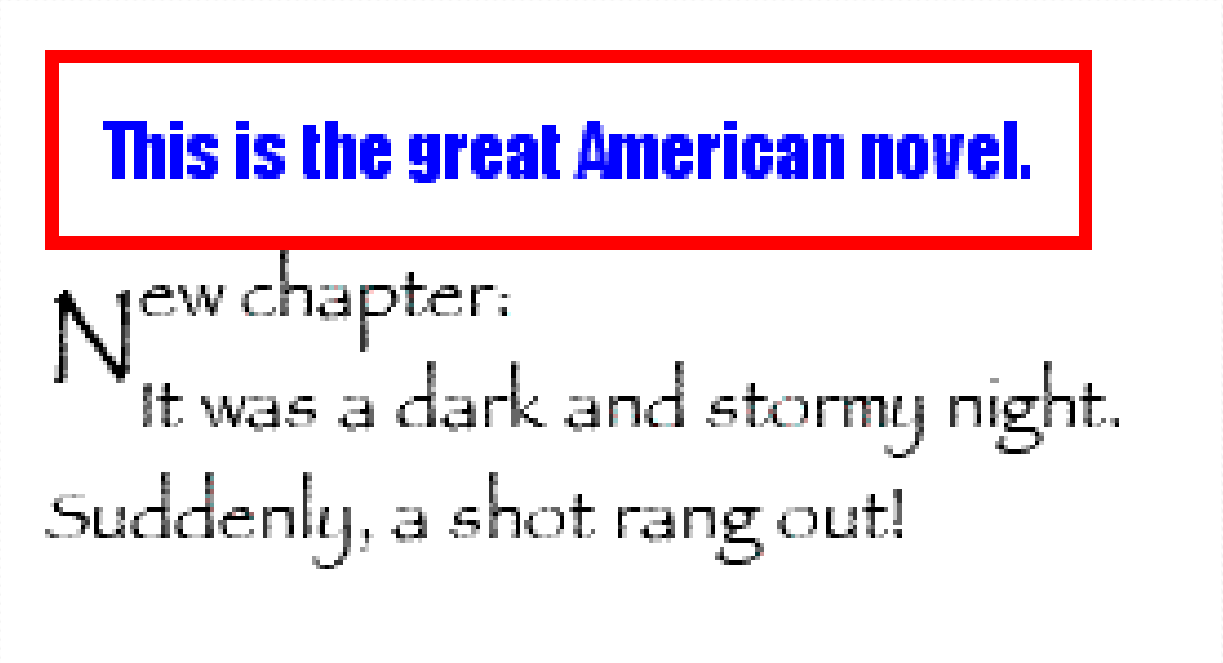
A novel example: XML

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE novel SYSTEM "novel.dtd">
<?xml-stylesheet href="styles.css" type="text/css"?>
<novel>
  <foreword>
    <paragraph>This is the great American novel.</paragraph>
  </foreword>
  <chapter>
    <paragraph>It was a dark and stormy night.</paragraph>
    <paragraph>Suddenly, a shot rang out!</paragraph>
  </chapter>
</novel>
```

A novel example: CSS

```
chapter {font-family: "Papyrus", fantasy}
foreword > paragraph {border: solid red; padding: 10px}
novel > foreword {font-family: Impact; color: blue}
chapter {display: block}
chapter:first-letter {font-size: 200%; float: left}
paragraph {display: block}
chapter:before {content: "New chapter: "}
```

A novel example: Result



- This is from Netscape 6.2--other browsers give different (not as good) results

Some font properties and values

- font-family:
  - inherit (same as parent)
  - Verdana, "Courier New", … (if the font is on the client computer)
  - serif | sans-serif | cursive | fantasy | monospace
    (Generic: your browser decides which font to use)
- font-size:
  - inherit | smaller | larger | xx-small | x-small | small | medium | large | x-large | xx-large | *12*pt
- font-weight:
  - normal | bold |bolder | lighter | 100 | 200 | … | 700
- font-style:
  - normal | italic | oblique

Shorthand properties

- Often, many properties can be combined:

```
h2 { font-weight: bold; font-variant: small-caps; font-size: 12pt; line-height: 14pt; font-family: sans-serif }
```

can be written as:

```
h2 { font: bold small-caps 12pt/14pt sans-serif }
```

Colors and lengths

- color: and background-color:
  - aqua | black | blue | fuchsia | gray | green | lime | maroon | navy | olive | purple | red | silver | teal | white | #FF0000 | #F00 | rgb(255, 0, 0) | ***Additional browser-specific names (not recommended)***
- These are used in measurements:
  - em, ex, px, %
    - font size, x-height, pixels, percent of inherited size
  - in, cm, mm, pt, pc
    - inches, centimeters, millimeters, points (1/72 of an inch), picas (1 pica = 12 points), relative to the inherited value

Some text properties and values

- text-align:
  - left | right | center | justify
- text-decoration:
  - none | underline | overline | line-through
- text-transform:
  - none | capitalize | uppercase | lowercase
- text-indent
  - ***length*** | ***10%*** (indents the first line of text)
- white-space:
  - normal | pre | nowrap

Pseudo-classes

- Pseudo-classes are elements whose state (and appearance) may change over time
- Syntax: ***element*:*pseudo-class*** {...}
  - :link
    - a link which has not been visited
  - :visited
    - a link which has been visited
  - :active
    - a link which is currently being clicked
  - :hover
    - a link which the mouse is over (but not clicked)
- Pseudo-classes are allowed anywhere in CSS selectors
- Note, however, that XML doesn't really support hyperlinks yet

Choosing good names

- CSS is designed to *separate content from style*
  - Therefore, names that will be used in HTML or (especially) in XML should describe *content, not style*
- Example:
  - Suppose you define `span.huge {font-size: 36pt}` and you use `<span class="huge">` throughout a large number of documents
  - Now you discover your users hate this, so you change the CSS to be `span.huge {font-color: red}`
  - Your name is inappropriate; do you change all your documents?
  - If you had started with `span.important {font-size: 36pt}`, your documents wouldn't look so dumb

References

- Some of the examples in this presentation were taken from the **W3Schools** online tutorial at http://www.w3schools.com/css/css_syntax.asp
- Dave Raggett's **Adding a Touch of Style** is a very nice online tutorial at http://www.w3.org/MarkUp/Guide/Style
- **Index DOT Css** has also been a great source of information about CSS: http://www.blooberry.com/indexdot/css/index.html
  - In particular, there is a list of when CSS features were first supported by which browsers (-- means "not yet supported") at http://www.blooberry.com/indexdot/css/supportkey/syntax.htm