3.1 Introduction

- The CSS1 specification was developed in 1996

- CSS2 was released in 1998
  - CSS2.1 reflects browser implementations

- CSS3 is partially finished and parts are implemented in current browsers

- CSSs provide the means to control and change presentation of HTML documents

- CSS is not technically HTML, but can be embedded in HTML documents

- A style sheet is a syntactic mechanism for specifying style information

- Style sheets allow you to impose a standard style on a whole document, or even a whole collection of documents

- Style is specified for a tag by the values of its properties

3.2 Levels of Style Sheets

- There are three levels of style sheets

  1. Inline - specified for a specific occurrence of a tag and apply only to that tag

  2. Document-level style sheets - apply to the whole document in which they appear

  3. External style sheets - can be applied to any number of documents

- When more than one style sheet applies to a specific tag in a document, the lowest level style sheet has precedence

  - In a sense, the browser searches for a style property spec, starting with inline, until it finds one (or there isn't one)

3.2 Levels of Style Sheets (continued)

- Inline style sheets appear in the tag itself

- Document-level style sheets appear in the head of the document

- External style sheets are in separate files, potentially on any server on the Internet

  - Written as text files with the MIME type `text/css`

  - A `<link>` tag is used to specify that the browser is to fetch and use an external style sheet file

```
<link rel = "stylesheet"  type = "text/css"
 href = "http://www.wherever.org/termpaper.css">
</link>
```

- External style sheets can be validated

  `http://jigsaw.w3.org/css-validator/`

3.3 Style Specification Formats

- Format depends on the level of the style sheet

- *Inline*:
  - Style sheet appears as the value of the `style` attribute
  - General form:

```
style = "property_1 : value_1 ;
         property_2 : value_2 ;
         ...
         property_n : value_n"
```

- *Document-level*:

  - Style sheet appears as a list of rules that are the content of a `<style>` tag

  - The `<style>` tag must include the `type` attribute, set to `"text/css"`

3.3 Style Specification Formats (continued)

- General form:

```
<style type = "text/css">
  rule list
</style>
```

- Form of the rules:
  selector { list of property/values }

  - Each property/value pair has the form:
    property: value

  - Pairs are separated by semicolons, just as in the value of a `<style>` tag

- Comments in the rule list must have a different form - use C comments (`/*...*/`)

- *External style sheets*

  - Form is a list of style rules, as in the content of a `<style>` tag for document-level style sheets

3.4 Selector Forms

1. *Simple Selector Forms*

- The selector is a tag name or a list of tag names, separated by commas
  - Examples:

```
h1, h3
p
```

2. *Class Selectors*

- Used to allow different occurrences of the same tag to use different style specifications

- A style class has a name, which is attached to a tag name

- For example,

```
p.narrow {property/value list}
p.wide {property/value list}
```


3.4 Selector Forms (continued)

2. *Class Selectors* (continued)

- The class you want on a particular occurrence of a tag is specified with the `class` attribute of the tag

- For example,

```
<p class = "narrow">
...
</p>
...
<p class = "wide">
...
</p>
```

3. *Generic Selectors*

- A generic class can be defined if you want a style to apply to more than one kind of tag

  - A generic class must be named, and the name must begin with a period

3.4 Selector Forms (continued)

3. *Generic Selectors* (continued)

- Example,

```
.sale { … }
```

- Use it as if it were a normal style class

```
<h1 class = "sale"> Weekend Sale </h1>
 ...
<p class = "sale"> … </p>
```

4. `id` *Selectors*

- An `id` selector allows the application of a style to one specific element

- General form:
  #specific-id {property-value list}

- Example:

```
#section14 {...}
```


3.4 Selector Forms (continued)

5. *Contextual Selectors*

- *Descendant Selectors*

`ul ol` – applies to `ol` when it is in a `ul` element

- *Child Selectors*

`ul > ol` – applies to `ol` when it is a child of a `ul` element

`p > h1 > em` – applies to `em` when it is the child of an `h1` element that is the child of a `p` element

`p:first-child, p:last-child, p:only-child`
for specific children

`p:empty`
for no children

3.4 Selector Forms (continued)

6. *Pseudo Classes*

- Pseudo classes are styles that apply when something happens, rather than because the target element simply exists

  - Names begin with colons

- `hover` classes apply when the mouse cursor is over the element

- `focus` classes apply when an element has focus

- `link` classes apply when a link has not been selected

- `visited` classes apply when a link previously has been selected

7. Universal Selector

```
* {color: red;}
```

- Applies to all elements in the document

3.5 Property Value Forms

- *There are 60 different properties in 7 categories*:
  - Fonts
  - Lists
  - Alignment of text
  - Margins
  - Colors
  - Backgrounds
  - Borders

- *Property Value Forms*

  - *Keywords* - `left`, `small`, ...
    - Not case sensitive

  - *Length* - numbers, maybe with decimal points
    - Units:
      - `px` - pixels
      - `in` - inches
      - `cm` - centimeters
      - `mm` - millimeters
      - `pt` - points
      - `pc` - picas (12 points)
      - `em` - height of the letter 'm'
      - `ex` - height of the letter 'x'

      - No space is allowed between the number and the unit specification
        e.g., `1.5 in` is illegal!

3.5 Property Value Forms (continued)

- Percentage - just a number followed immediately by a percent sign

- URL values
  - `url(`protocol`://`server`/`pathname`)`

- Colors
  - Color name
  - `rgb(`n1`,` n2`,` n3`)`
    - Numbers can be decimal or percentages
  - Hex form: `#XXXXXX`

- Property values are inherited by all nested tags, unless overriden

3.6 Font Properties

- `font-family`
  - Value is a list of font names - browser uses the first in the list it has

  - `font-family: Arial, Helvetica, Futura`

  - Generic fonts: `serif`, `sans-serif`, `cursive`, `fantasy`, and `monospace` (defined in CSS)
    - Browser has a specific font for each
  - If a font name has more than one word, it should be single-quoted

3.6 Font Properties (continued)

- `font-size`
  - Possible values: a length number or a name, such as `smaller`, `xx-large`, etc.
  - Points or picas do not display the same
  - Percentages and `em` are the best
- Font variants
  - Default is `normal`, but can be set to `small-caps`
- `font-style`
  - `italic`, `oblique` (useless), `normal`
- `font-weight` - degrees of boldness
  - `bolder`, `lighter`, `bold`, `normal`
    - Could specify as a multiple of 100 (100 – 900)
- `font` (shorthand)
  - For specifying a list of font properties

```
font: bolder 14pt Arial Helvetica
```

  - Order must be: style, weight, size, name(s)

3.6 Font Properties (continued)

→ SHOW `fonts.html` and display
→ SHOW `fonts2.html` and `styles.css` and display

- The `text-decoration` property
  - `line-through`, `overline`, `underline`, `none`

→ SHOW `decoration.html` & display

- `letter-spacing` – value is any length property value

- *Text Spacing*

  - `letter-spacing` property – the amount of space between the letters in words – *tracking*
    Possible values: `normal` or any length value
    - Positive length values increase spacing
    - Negative length values decrease spacing

  - `word-spacing` property – the amount of space between words
    Possible values – like those of `letter-spacing`

  - `line-height` property – space between lines – *leading*
    Possible values – a number, which is the number of times the font size, or a percentage

→ SHOW `text_space.html` and display

3.7 *List properties*

- `list-style-type`

  - *Unordered lists*
    - Bullet can be a disc (default), a square, or a circle

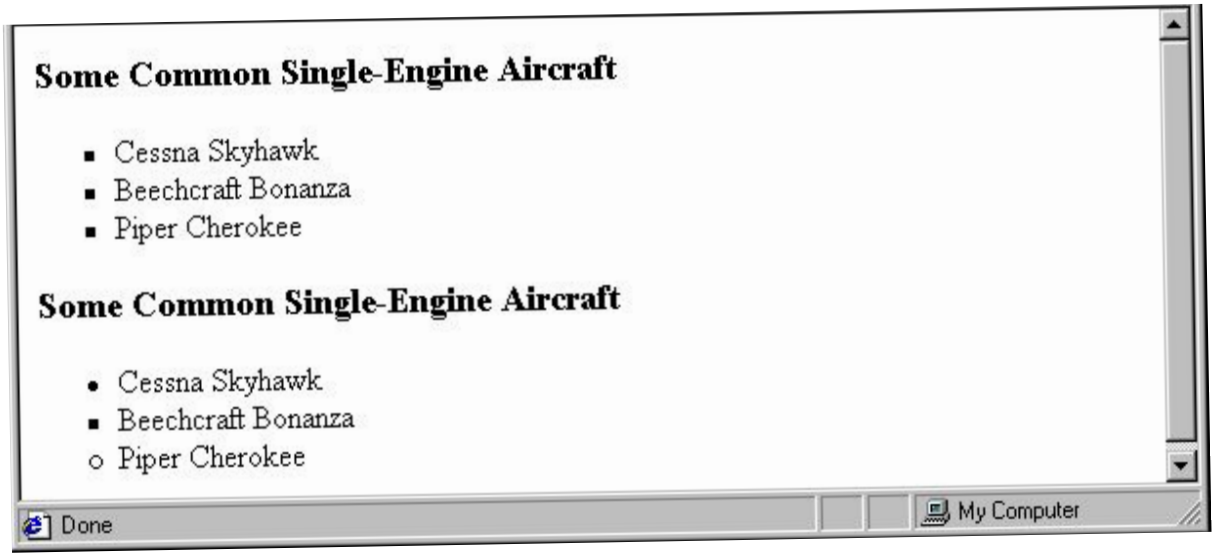
    - Set it on either the `<ul>` or `<li>` tag
      - On `<ul>`, it applies to all items in the list

```
<h3> Some Common Single-Engine Aircraft </h3>
 <ul style = "list-style-type: square">
     <li> Cessna Skyhawk </li>
     <li> Beechcraft Bonanza </li>
     <li> Piper Cherokee </li>
 </ul>
```

- On `<li>`, `list-style-type` applies to just that item

```
<h3> Some Common Single-Engine Aircraft </h3>
<ul>
    <li style = "list-style-type: disc">
         Cessna Skyhawk </li>
    <li style = "list-style-type: square">
         Beechcraft Bonanza </li>
    <li style = "list-style-type: circle">
         Piper Cherokee </li>
</ul>
```


3.7 *List properties* (continued)



- Could use an image for the bullets in an unordered list

- Example:

```
<li style = "list-style-image:
             url(bird.jpg)">
```


3.7 *List properties* (continued)

- ***On ordered lists*** - `list-style-type` **can be used to change the sequence values**

Property value	*Sequence type*	*First four*
`decimal`	Arabic numerals	1, 2, 3, 4
`upper-alpha`	Uc letters	A, B, C, D
`lower-alpha`	Lc letters	a, b, c, d
`upper-roman`	Uc Roman	I, II, III, IV
`lower-roman`	Lc Roman	i, ii, iii, iv

  - **There are several more, including** `none`

→ **SHOW** `sequence_types.html` **and display**

3.8 Alignment of Text

- The `text-indent` property allows indentation
  - Takes either a length or a % value
- The `text-align` property has the possible values, `left` (the default), `center`, `right`, or `justify`
- Sometimes we want text to flow around another element - the `float` property
  - The `float` property has the possible values, `left`, `right`, and `none` (the default)
  - If we have an element we want on the right, with text flowing on its left, we use the default `text-align` value (`left`) for the text and the `right` value for `float` on the element we want on the right

3.8 Alignment of Text (continued)

```
<img src = "c210.jpg"
          style = "float: right" />
```

-- Some text with the default alignment - `left`

3.9 *Colors*

- There are three color collections

1. There is a set of 17 colors that are guaranteed to be displayable by all graphical browsers on all color monitors

2. There are 147 named colors – see Appx. B

3. There is a larger set, the *Web Palette*

- 216 colors
- Use hex color values of `00`, `33`, `66`, `99`, `CC`, and `FF`

3.8 *Colors* (continued)

- The `color` property specifies the foreground color of elements

```
<style type = "text/css" >
  th.red {color: red}
  th.orange {color: orange}
</style>
…
<table>
  <tr>
    <th class = "red"> Apple </th>
    <th class = "orange"> Orange </th>
    <th class = "orange"> Screwdriver </th>
  </tr>
</table>
```

- The `background-color` property specifies the background color of elements

→ SHOW `back_color.html` and display

3.10 The Box Model

- Borders – every element has a `border-style` property
  - Controls whether the element has a border and if so, the style of the border
  - `border-style` values: `none`, `dotted`, `dashed`, and `double`
  - `border-width` – `thin`, `medium` (default), `thick`, or a length value in pixels
  - Border width can be specified for any of the four borders (e.g., `border-top-width`)
  - `border-color` – any color
  - Border color can be specified for any of the four borders (e.g., `border-top-color`)
  - Table borders and table cell borders
    - To get cell borders:

```
td, th {border: thin solid black}
```

    - To get table borders:

```
table {border: thin solid black}
```

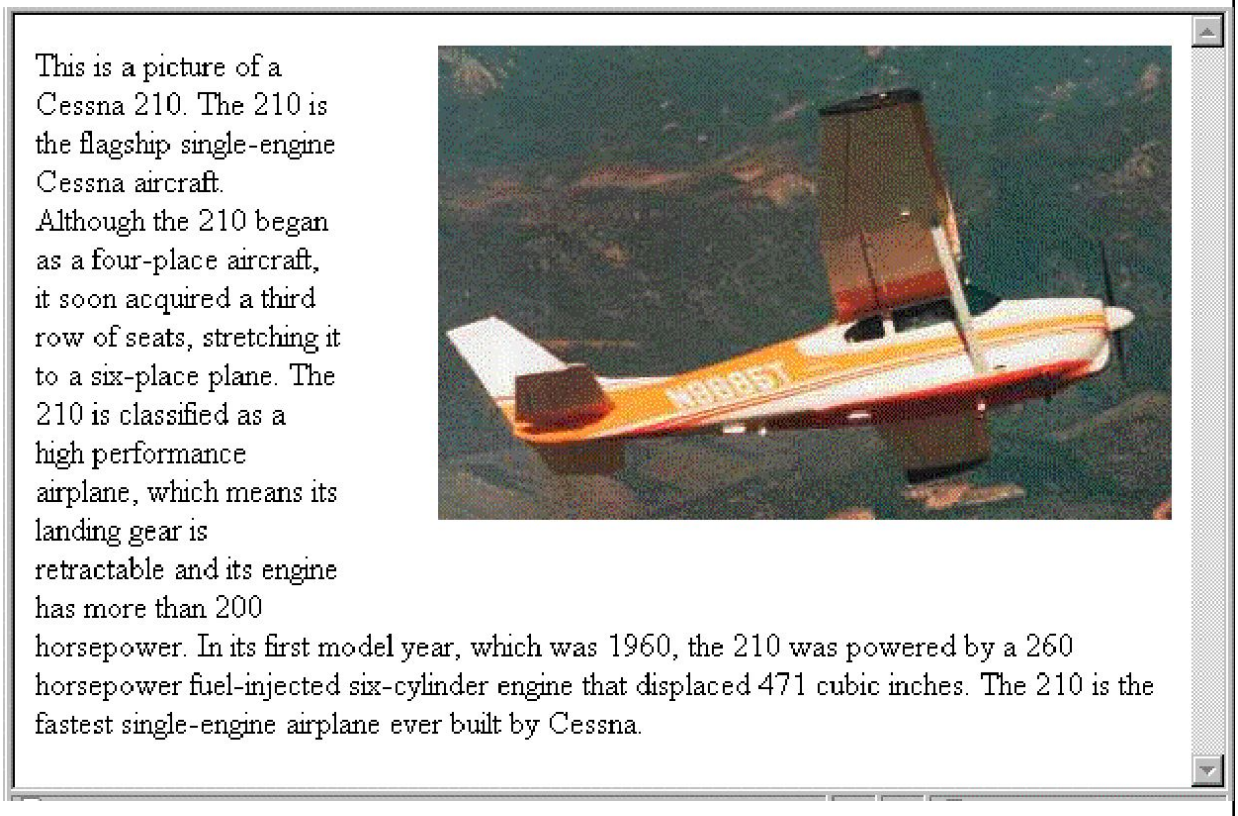
→ SHOW `borders.html` and display

3.10 The Box Model (continued)

- Margin – the space between the border of an element and its neighbor element

  - The margins around an element can be set with `margin-left`, etc. - just assign them a length value

```
<img src = "c210.jpg " style = "float: right;
   margin-left: 0.35in;
   margin-bottom: 0.35in" />
```



3.10 The Box Model (continued)

- Padding – the distance between the content of an element and its border

  - Controlled by `padding`, `padding-left`, etc.

→ SHOW `marpads.html` and display

3.11 Background Images

- The `background-image` property

→ SHOW `back_image.html` and display

- Repetition can be controlled

  - `background-repeat` property

    - Possible values: `repeat` (default), `no-repeat`, `repeat-x`, or `repeat-y`

  - `background-position` property

    - Possible values: `top`, `center`, `bottom`, `left`, or `right`

3.12 The and tags

- One problem with the font properties is that they apply to whole elements, which are often too large

  - Solution: a new tag to define an element in the content of a larger element -

  - The default meaning of is to leave the content as it is

```
<p>
Now is the <span> best time </span> ever!
</p>
```

  - Use to apply a document style sheet to its content

```
<style type = "text/css">
  .bigred {font-size: 24pt;
           font-family: Ariel; color: red}
</style>

...

<p>
   Now is the
       <span class = "bigred">
   best time </span> ever!
</p>
```


3.12 The and tags (continued)

- The tag is similar to other HTML tags, they can be nested and they have id and class attributes

- Another tag that is useful for style specifications:

  - Used to create document sections (or divisions) for which style can be specified

    - e.g., A section of five paragraphs for which you want some particular style

3.13 Conflict Resolution

- A conflict occurs when there are two or more values for the same property on the same element

- *Sources of conflict*:

  1. Conflicting values between levels of style sheets

  2. Within one style sheet

  3. Inheritance can cause conflicts

  4. Property values can come from style sheets written by the document author, the browser user, and the browser defaults

- *Resolution mechanisms*:

  1. Precedence rules for the different levels of style sheets

  2. Source of the property value

  3. The specificity of the selector used to set the property value

  4. Property value specifications can be marked to indicate their weight (importance)

3.12 Conflict Resolution (continued)

- Weight is assigned to a property value by attaching `!important` to the value

- Conflict resolution is a multistage process, called *the cascade*:

  1. Gather all of the style specs from the different levels of style sheets

  2. All available specs, from all sources, are sorted by origin and weight, using the following rules, which are given in precedence order:

     a. Important declarations with user origin

     b. Important declarations with author origin

     c. Normal declarations with author origin

     d. Normal declarations with user origin

     e. Any declarations with browser (or other user agent) origin

3.12 Conflict Resolution (continued)

3. If any conflicts remain, sort them by specificity:

   a. `id` selectors

   b. Class and pseudo-class selectors

   c. Contextual selectors

   d. Universal selectors

4. If there are still conflicts, resolve them by precedence to the most recently seen specification

