SIU CARBONDALE

SOUTHERN ILLINOIS UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CASCADING STYLE SHEETS

Introduction

- **Purpose**: for separate presentation control
- The CSS1 specification was developed in 1996
- CSS2 was released in 1998
- CSS3 is partially finished and implemented by browsers
- CSSs provide the means *to control and change* presentation of HTML documents
- CSS is not technically HTML, but embedded in HTML documents
- Style sheets allow you to impose a standard style on a whole document, or even a whole collection of documents
- Style is specified for tags by the values of its properties

Levels of Style Sheets

- There are three levels of style sheets
  - **Inline** - specified for a specific occurrence of a tag and apply only to that tag – This fine-grain style, which defeats the purpose of CSS – uniform style
  - **Document-level** style sheets - apply to the whole document in which they appear
  - **External style** sheets - can be applied to any number of documents
- **Conflict resolution**: When more than one style sheet applies to a specific tag, the ***lowest level*** style sheet has *precedence*
  - In a sense, the browser searches for a style property specification, starting with inline (the lowest) until it finds one (or there isn't one)

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 dwell in separate files, potentially on any server on the Internet
  - Written **as text files** with the **MIME type** `text/css`
  - Must be linked to from an html file using `<link>` such as

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

  - External style sheets can also be **validated** as well at

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

    But usually you don't need to do so!

Style Specification Formats: Inline

- Format depends on the level of the style sheets
- Inline:
  - Style sheet appears as the value of the **`style`** attribute of a tag
  - General form:

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

    - Example

```
<ul>
  <li STYLE="font-size: x-large; font-style: italic"> 'A Toast To My Toaster'
```

Style Format: Document-level

- Style sheet is **a list of rules** that are content of a `<style>` tag
- The `<style>` tag must include the `type` attribute, set to `"text/css"`
- The list of rules **should** be placed in an HTML comment, because it is **not** HTML
- Comments within the list of the rules must have a different form – usually use the **C** comments (`/*…*/`)

Style Format: Document Level, cont'

- General form:
- `<style type = "text/css">`
- `<!--`
- rule list
- `-->`
- `</style>`
- Form of the rules:
- selector {list of property/values pairs}
  - Each property/value pair has the form:
  - *property: value*
  - Pairs are separated by semicolons ";"

Style Format: External style sheets

- A list of style rules, ***just as in*** the content of a **<style>** tag for a document-level style sheet
- But reside in a separate file with the extension name ***.css***
- Of course, the **css** file then needs be linked to the web document that uses it (we'll show example of this later)

Simple Selector Forms

- The **selector** is a tag name or a list of tag names, separated by commas, e.g.,

  p

  h1, h3

- *Contextual selectors, such as*

  ol ol li /* describe a hierarchy, applies only to li */

- Examples:

```
h2, h3 {font-size: 40pt;}    /* applies to both */

body b em {font-size: 40pt;}      /* only to em in the hierarchy.*/
```

Class Selectors

- 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
  - `p.narrow {`property/value list`}`
  - `p.wide {`property/value list`}`
- The class you want on a particular occurrence of a tag is specified with the **`class`** attribute of the tag
- Use example,

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

Generic Selectors

- A generic class applies to more than one kind of tags
- A generic class must be named and begin with a period
- Example,
- .really-big { … }
- Use it as if it were a normal style class, e.g.,
- `<h1 class = "really-big"> … </h1>`
- `…`
- `<p class = "really-big"> … </p>`

The `id` Selectors

- An **`id`** selector allows the application of a style to one specific element (instead of a tag; what's the difference?)

- General form:

  #specific-id {property-value list}

- Example:

```
#section14 {font-size: 20}
…
<h2 id="section14"> my heading </h2>

/* the id must be defined for the element (via its id
  attribute) as in this example*/
```

Universal Selectors

- Start with "*" and apply to **all** elements in a document, e.g.,

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

Pseudo Classes

- Pseudo classes are style classes that apply when something happens, rather than something simply exists
- The class names begin with colons
- Two basic pseudo classes:
  1. `Tag-name:hover` classes apply when the mouse cursor is over the element
  2. `Tag-name:focus` classes apply when an element gets focus
- More pseudo classes:
  - a:link {color:green;} a:visited {color:green;} a:active {color:yellow;}
- Examples (next slide)

Pseudo Class Example

```
<!-- pseudo.html -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Checkboxes </title>
    <style type = "text/css">
      input:hover {color: red;}
      input:focus {color: green;}
    </style>
  </head>
  <body>
    <form action = "">
      <p>
        Your name:
        <input type = "text" />
      </p>
    </form>
  </body>
 </html>
```

Properties

- There are 60 different style 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 - height of the letter ‘x’
  - No space is allowed between the number and the unit specification
    - e.g., 1.5 in is illegal!

Property Value (continued)

- Percentage -- a number followed by a percent sign, e.g. 8%
- URL values
  - url(protocol://server/pathname)
- Values of Colors
  - Color name, e.g., red, blue
  - rgb(n1, n2, n3)
    - Numbers can be decimal or percentages
  - Hex form: #XXXXXX
- **Note**: property values are **inherited** by all **nested tags**, **unless overridden**

Font Properties

- **font-family**
  - Value is a list of font names - browser uses the first in the list it has, for example
    - font-family: Arial, Helvetica, Courier
  - Can also be generic fonts such as: 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 **must** be '**single-quoted**'

Font Properties (continued)

- font-size
  - Values: a length number or a name, e.g., smaller, xx-large, etc.
- font-style
  - italic, oblique, normal
- font-weight – degrees of boldness
  - bolder, lighter, bold, normal
    - Could specify as a **multiple of 100** (100 – 900)
  - **font** -- a ***shorthand*** specifying a **list** of many font properties, e.g.,

    font: italic, bolder 14pt Arial Helvetica

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

Font Properties (continued)

- The **`text-decoration`** property
  - `line-through`, `overline`, `underline`, `none`
  - `letter-spacing` – value can be any length property value

List properties

- The **`list-style-type`** property
- *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 list items

```
<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>
```

List properties (continued)

- On <li>, list-style-type applies to just that item, e.g.,

```
<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>
```

List properties (continued)

- Could also use an image for the bullets in an unordered list
  - Example:

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

- *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

- Note: CSS2 has more, like `lower-greek` and `hebrew`

Colors

- *Color is a problem for the Web for two reasons*:
  1. Monitors vary widely
  2. Browsers vary widely

- There is a set of 16 named colors that are guaranteed to all browsers:

black	000000	green	008000
silver	C0C0C0	lime	00FF00
gray	808080	olive	808000
white	FFFFFF	yellow	FFFF00
maroon	800000	navy	000080
red	FF0000	blue	0000FF
purple	800080	teal	008080
fuchia	FF00FF	aqua	00FFFF

- There 147 named colors
- There can be potentially 16 million different colors

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 border = "5">
   <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

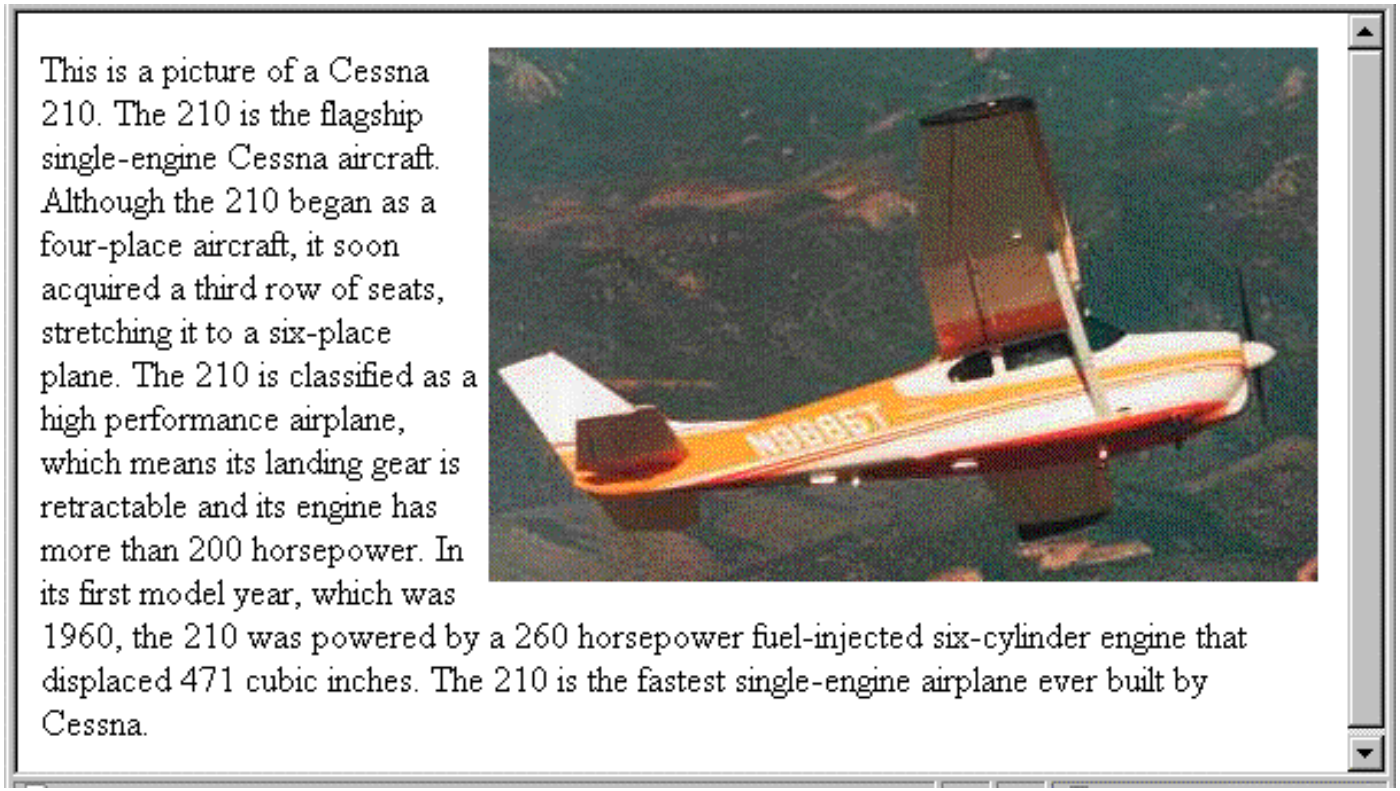
Alignment of Text

- 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 – by using the **`float`** property
  - The `float` property has the possible values, `left`, `right`, and `none` (the default)
  - If we have an element (on the right), we want text flowing on its left, we can use the `right` value for `float` on the element (on the right) and the default `text-align` value (`left`) for the text (on the left)
  - See the next example

Alignment of Text (continued) For Example

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

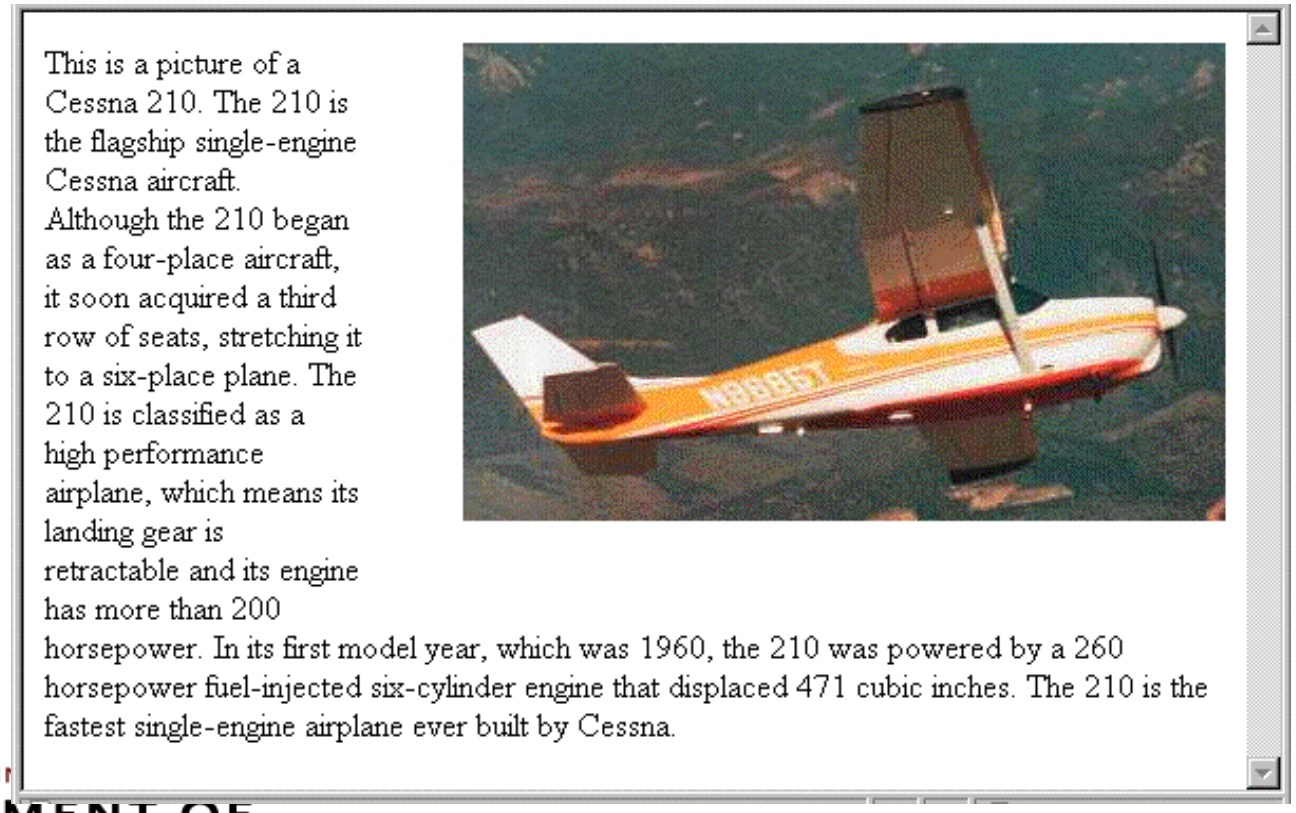
[text to float on the left …]



Alignment of Text (continued)

- Margin – the space between the border of an element and its neighbor elements
- The margins around an element can be set with **`margin-left`**, etc. -- just assign them a length value, for example,

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

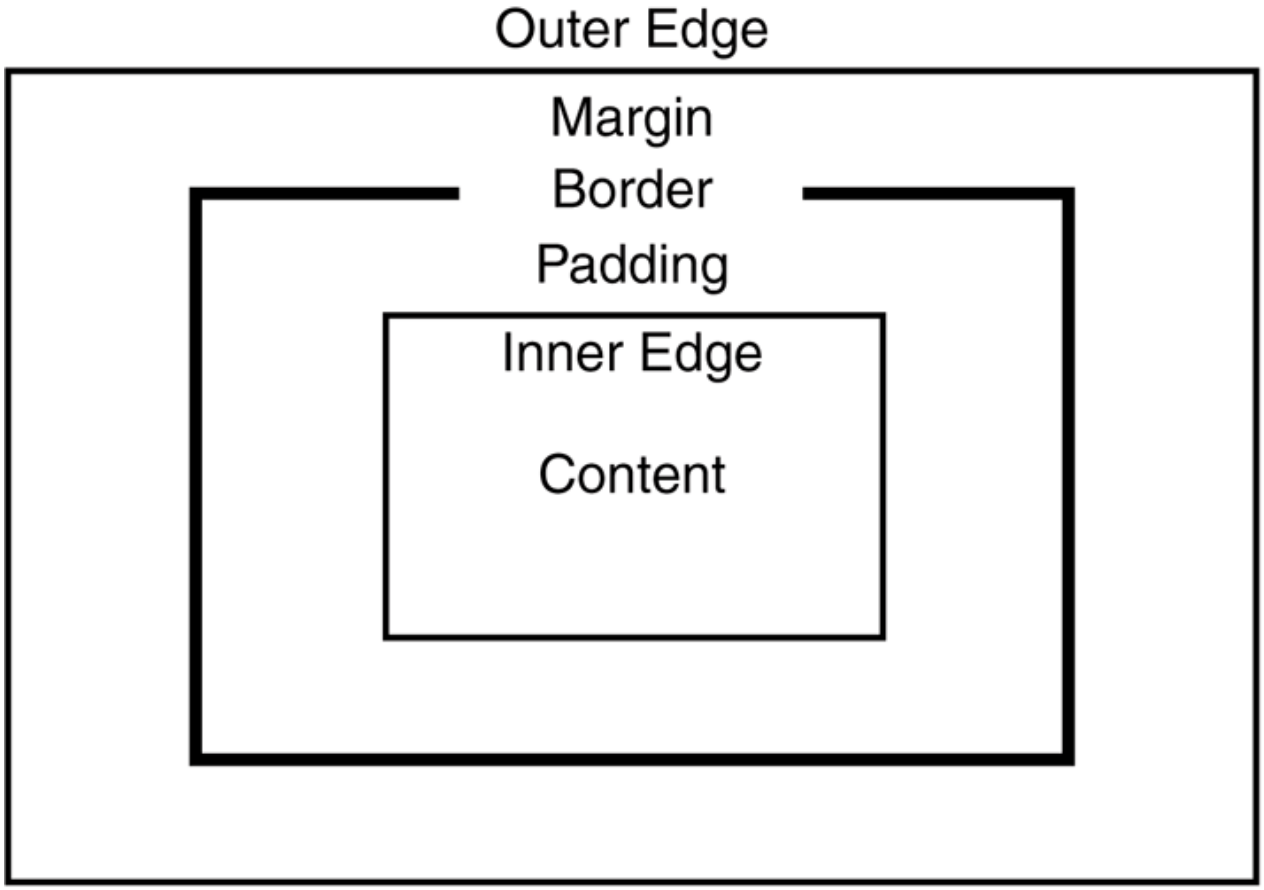


Alignment of Text (continued)

- Padding – the distance between the content of an element and **its** border
  - Controlled by `padding`, `padding-left`, etc.

The box model

Virtually all doc elements can have borders



Note: shown edges are imaginary and actually invisible !

The Box Model (continued)

- Borders – every element has a `border-style` property that 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, etc`)
  - `border-color` – any color
  - Border color can be specified for any of the four borders (e.g., `border-top-color, etc`)

Background Images

- The **`background-image`** property
- Repetition can be controlled by
  - **`background-repeat`** property (useful!)
  - Possible values: `repeat` (default), `no-repeat`, `repeat-x`, or `repeat-y`
- **The `background-position`** property
  - Possible values: `top`, `center`, `bottom`, `left`, or `right`

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

Conflict Resolution, cont'

- *Resolution mechanisms (you may use)*:

  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 carry weight (importance)
     - *Weight is assigned to a property value by attaching* !important *to the value, e.g.,*

       p.x {font-style: italic !important; font-size:20}

Conflict resolution – a multistage process

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:
   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. 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, use the "most recently seen" policy
5. "Trial and Error"