CSS Introduction

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

- CSS is not technically HTML, but can be 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 a tag by the values of its properties

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

     - This is fine-grain style, which defeats the purpose of style sheets - uniform style

  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)

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>
```

  - An alternative way to reference an external style sheet:

    `@import url(`filename`);`

    - Appears at the beginning of the content of a `style` element (later)

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"`

  - The list of rules must be placed in an HTML comment, because it is <u>not</u> HTML

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

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

- *External style sheets*

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

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
```

- ***Contextual selectors***

```
ol ol li
```

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}
```


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

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 allow the application of a style to one specific element

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

- Example:

```
#section14 {font-size: 20}
```

5. *Universal Selectors*

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

- Applies to all elements in the document

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

```
<!-- pseudo.html – Does not work with IE7 -->
<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>
```


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)

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

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

Font Properties (continued)

- If a font name has more than one word, it should be single-quoted

- `font-size`
  - Possible values: a length number or a name, such as `smaller`, `xx-large`, etc.
- 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`
  - For specifying a list of font properties

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

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

Font Properties (continued)

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

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

→ SHOW `decoration.html` & display

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

List properties

- `list-style-type`

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

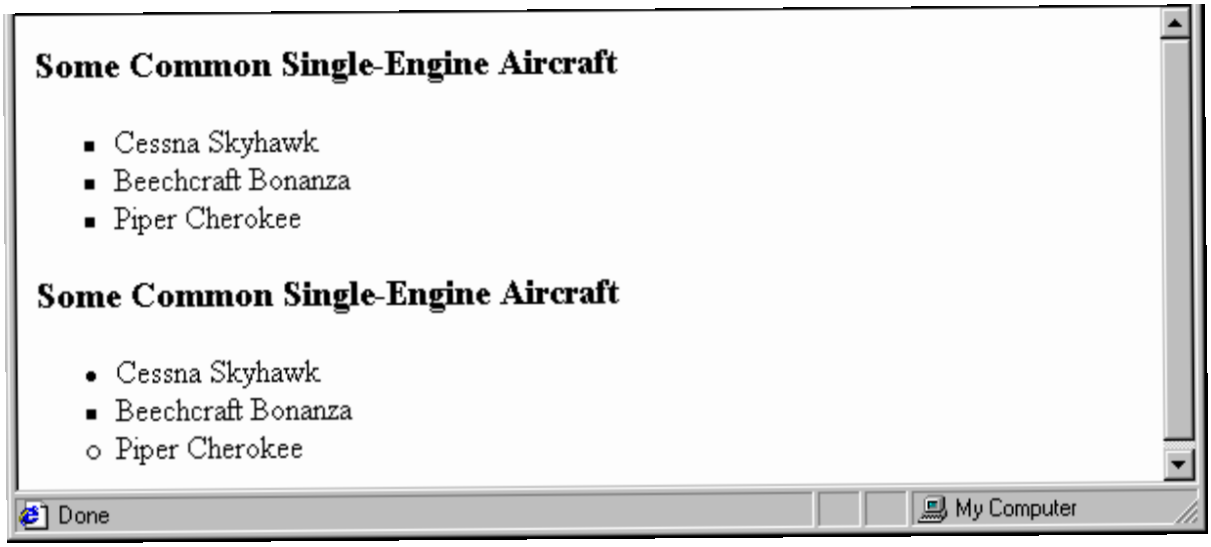
    - Set it on either the `<ul>` or `<li>` tag
      - On `<ul>`, it applies to list items

```
<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

```
<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 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

Alignment of Text (continued)

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

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

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>
```

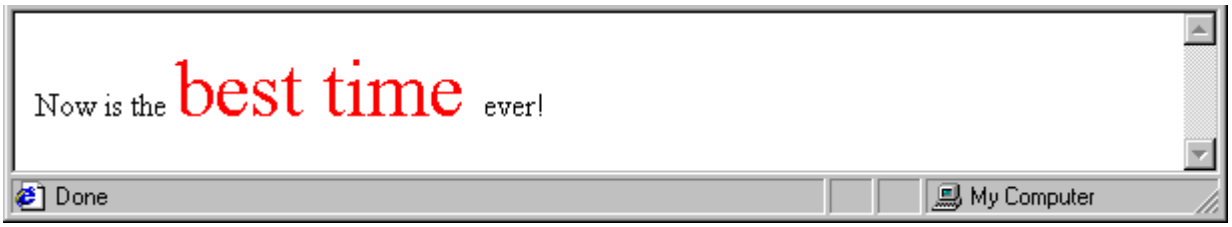
  - Use to apply a document style sheet to its content

```
<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>
```


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

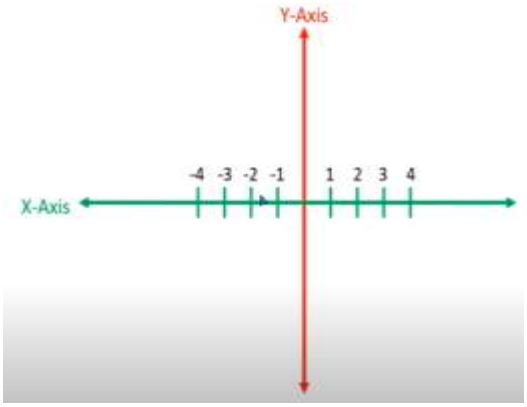
CSS Position:

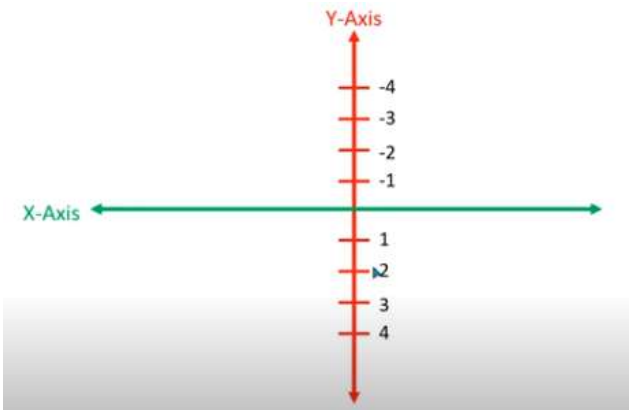
The position property specifies the type of positioning method used for an element.

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

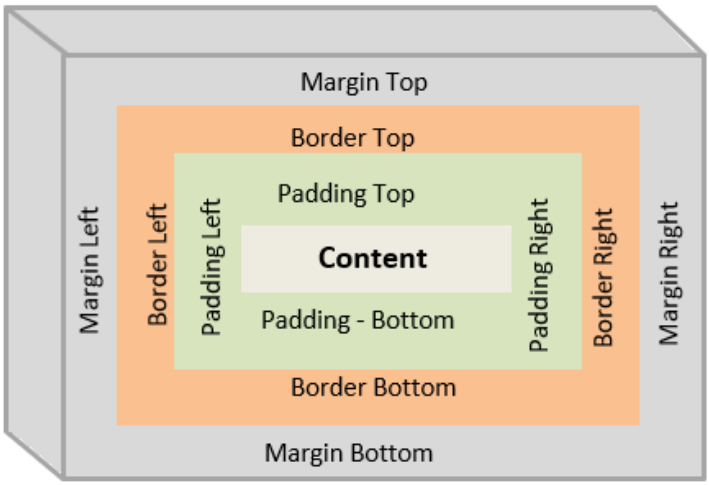
Elements are then positioned using the **top, bottom, left, and right** properties.





Ex: **position: relative;**
left:100px;
top:100px;

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

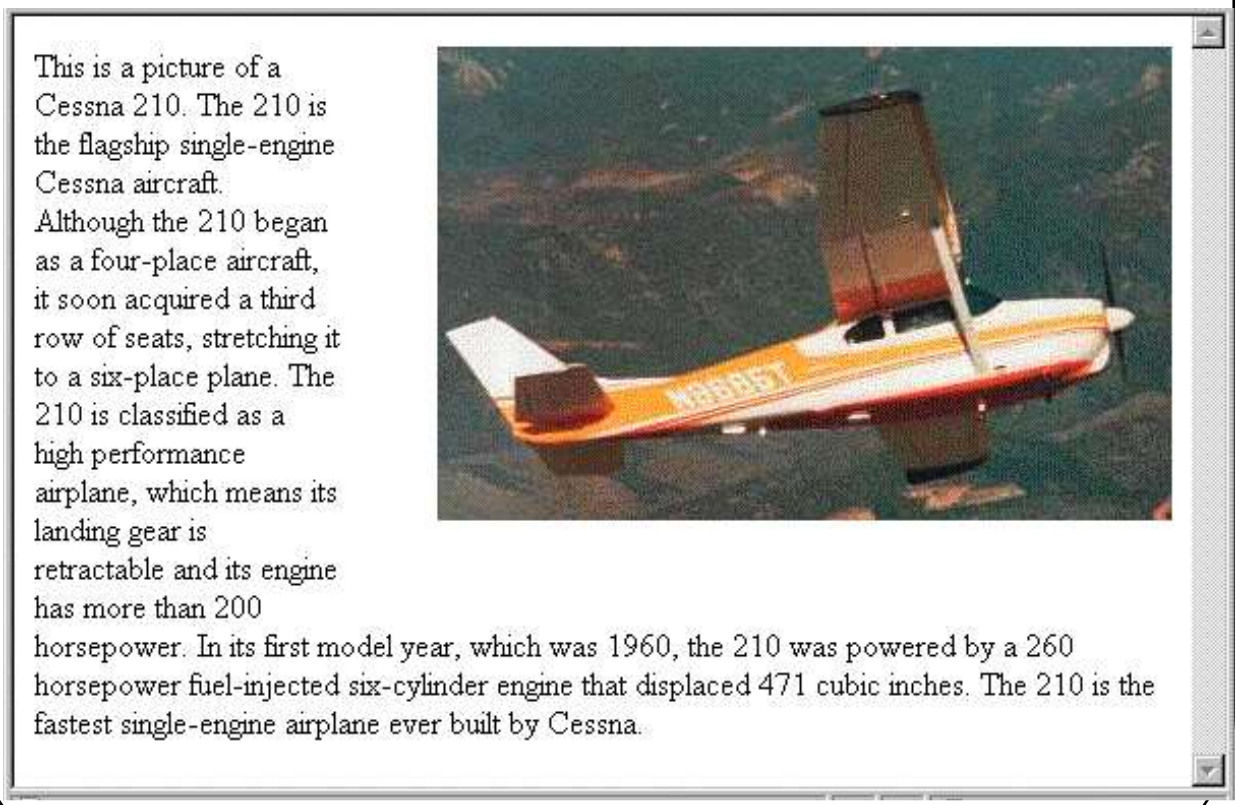
  - Border color can be specified for any of the four borders (e.g., `border-top-color`)

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" />
```



The Box Model (continued)

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

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

Background Images

- The `background-image` property

- 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`

Conflict Resolution

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

-There are 4 Cascading concepts for solving conflicts between CSS rules.

1. Origin
2. Merge
3. Inheritance
4. Specificity

1. Origin:
   Declarations are in conflicts, the last declaration wins.(Bottom most style will be applied)

2. Merge:

   Declaration are not in conflicts, they just merge together & all properties are applied.

3. Inheritance:
   Declarations are inherited from the Parent Tag.

Conflict Resolution (continued)

4. Specificity:

Specificity concept is based on the rules that the most specific combination of selectors Wins

**Ex1. RED TEXT **

Specificity:

Style	Id	Class,pseudo-class,attributr	No. of affected elements
1	0	0	0

Ex2.

```
p{
        Color: green;
}
```

Specificity:

Style	Id	Class,pseudo-class,attributr	No. of affected elements
0	0	0	1

Conflict Resolution (continued)

Ex 3.

```
 div #para{
Color: blue;
}
```

Specificity:

Style	Id	Class,pseudo-class,attribute	No. of affected elements
0	1	0	1

```
div.green p{
color: green;
}
```

Specificity:

Style	Id	Class,pseudo-class,attribute	No. of affected elements
0	0	1	2

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

Ex.

```
p{
        color: red !important;
}
```

