CSS

Chapter 3

- **Cascading Style Sheets (CSS)** - describes the way the contents should look including colors, font and alignment.
- Original HTML used tags to format text. These tags are deprecated, don't use them.
- Style sheets describe **how** the information is to be displayed.
- Style sheets can be implemented in three ways:
  - Inline with each element. ☹
  - Embedded in the head section as a style element. ☹
  - Placed in own CSS file. ☺
- Current version is CSS3. Most browsers support CSS 2.1 but CSS3 contains many new features. You can check browser support at http://www.w3schools.com/cssref/css3_browsersupport.asp
- You must write valid CSS for this course. You can verify your CSS at http://jigsaw.w3.org/css-validator/

- **Rule** - unit of CSS that specifies a set of page elements or tags and a set of styles to apply to them.
- **Selector** - describes which content a rule applies to.
- A general CSS rule looks like:

```
selector1, selector2, . . . , selectorN {
    property: value;
    property: value;
    . . .
    property: value;
}
```

- Property names are always lowercase.
- Multi-word properties are separated by hyphens: font-family.
- The **media** attribute allows the page to apply different styles depending on the browser.
- CSS comments are C-like: /* comments */
- Details for all CSS names, properties and values can be found at http://www.w3schools.com/cssref/.

In order to apply CSS to an html page, it must be linked in the head section. If you have a style.css file, this is how you would link to it:

```
    <link href="style.css" type="text/css" rel="stylesheet" />
```

Example

The following css file makes all pararaphs and h2 headers red:

```
p, h2{
    color: red;
}
```

Example

```
p {
    color: red;
    background-color: yellow;
}
or:
p {
    color: rgb(255,0,0);
    background-color: rgba(255,255,0,0);
}
```

Can also use hue-saturation-luminance color codes or hexadecimal color codes.

Simple color schemes are preferred.

Light font color with dark backgrounds generally look horrible.

Example

```
p {
   color: red;
   font-family: "Courier New";
}
```

- Multi-word font names must be in quotes.
- **Monospace** - font where all letters are drawn with the same width.
- Fonts may not be installed on a client's machine so you should declare more than one:
  font-family: "Garamound", "Times New Roman", cursive;
- Generic fonts on all machines: cursive, fantasy, monospace, sans-serif, serif.

Example

```
p {
    color: red;
    font-family: “Courier New”;
    font-size: 15pt;
}
```

Font sizes can be specified by:

- pixels (px)
- points (pt)
- m-sizes (em)
- percentages (150%)
- absolute (small, xx-large, medium. . .)
- relative (smaller, larger)

Example

```
p {
    color: red;
    font-family: "Courier New";
    font-size: 15pt;
    font-weight: bold;
    font-style: italic;
}
```

```
p{
    font: italic bold 15pt "Courier New";
    color: red;
}
```

- The properties that can be set by font are, in order:
  font-style font-variant font-weight font-size/line-height font-family.
- Can include and specify your own font using @font-face:
  http://www.w3schools.com/cssref/css3_pr_font-face_rule.asp

- There are many text properties that can affect the appearance of text. You can:
  - Change the alignment.
  - Underline, strike-out or cause text to blink.
  - Modify indentation.
  - Change the vertical size of each line.
  - ...
- Blinking text is obnoxious, don't use it.

Example

```
p {
   text-align: right;
   text-decoration: underline;
   text-indent: 5pt;
}
strong {
   text-decoration: overline;
}
```

- You can specify the way the background looks.
    - Add a background color.
    - Add a background image.
    - Specify how a background image is displayed (repeated, stretched, positioning, etc).

Example

```
body {
   background-image: url(background.png);
   background-size: cover;
   background-repeat: no-repeat;
}
```

- You can specify the way different lists look.
  - Use Roman numerals.
  - Use Greek letters.
  - Change the circle to being filled/unfilled.
  - ...

Example

```
ol {
   list-style-type: lower-greek;
}
```

You can specify the way tables look by changing the fonts of each row, heading or column.

Example

```
table {
    caption-side: bottom;
    font-size: 12pt;
}
caption {
    font-size: 18pt;
}
tr {
    color: green;
}
th {
    font-style: italic;
    background-color: yellow;
    text-align: right;
}
```

Changing a tr font is easy. However, changing a columns font requires a bit more work. One option is to create classes.

Example

```
.states {
    color: green;
    background-color: yellow;
    font-size: 12pt;
}
.capitals {
    color: blue;
    background-color: red;
    font-size: 15pt;
}
```

Using the classes from the previous slide, our html table could look like this.

Example

```
<table>
   <col class="states" />
   <col class="capitals" />
...
</table>
OR:
<table>
   <colgroup class="states" />
   <col />/* each column uses the state format */
   <col />
...
</table>
```

- CSS properties are *inherited* from the outer element to the inner one.
- Common to apply style to body.
- If there are conflicts, in general the more specific selector is chosen.
- If there are multiple selectors of the same name, the last rule listed is chosen.

Example

```
p {
   color: red;
   text-decoration: underline;
   text-indent: 5pt;
}
strong {
   text-decoration: overline;
   color: purple;
}
```

- It's easy to apply a style to all occurrences of a particular element.
- We can give HTML elements an *id*. An id must begin with a letter and must be unique throughout the page.

Example

In the CSS file:

```
#someID {
   color: red;
   text-decoration: underline;
   text-indent: 5pt;
}
```

In the HTML file:

```
<p id="someID">This is my red and underlined paragraph that is indented by 5 points.</p>
```

- ID's are useful if you have unique elements you want special styles applied to. However, if you have many elements with the same style, ID's are not good.
- The *class* attribute is useful here. Multiple elements can have the same class.
- **Class selector** - using CSS to apply style to all elements of a class.

Example

In the CSS file:

```
.myClass {
    color: red;
}
p {
    color: blue;
    font-size: 18pt;
}
```

In the HTML file:

```
<p class="myClass">Red Stuff</p>
<p>Blue Stuff</p>
<p class="myClass">Red Stuff</p>
```

- An element can have more than 1 class. Each class is separated by a space. For example:
  <p class="classOne classTwo classThree">My text!</p>
- CSS allows you to state a tag name before a class name. In the CSS file:
  ```
  h1.myClass {
      color: red;
  }
  ```
- Use good names when choosing id and class names. a, b and c are horrible names.

- Pseudo-classes target an element under specific conditions.
- Can target clicked links, unclicked links, every other row in a table, etc.
- A pseudo-class selector written by itself applies to all elements.

Example

```
a:hover {
    background-color: yellow;
    font-style: italic;
}
li:nth-child(odd) {
    background-color: gray;
    font-weight: bold;
}
```