1 ☐ **FORMATTING WEB PAGES WITH CSS**

HTML5 and CSS3

2 ☐ **STYLES IN HTML DOCUMENTS**

- Styles provide a method of creating consistent formatting of elements throughout an entire website
- It is much simpler to format all the headings, paragraphs, list elements, etc. for *all the pages* in the site so that they are identical in style
- The style is *defined once* and *applied automatically* each time the tag and/or element is used

3 ☐ **STYLE IMPLEMENTATIONS**

- Style sheets may be applied at three different levels:
  - Linked—style sheet is created as a *separate document file* and linked to each web page that implements the style sheet
  - Embedded—style sheet is placed into a single web page and implemented only in that document
  - Inline—placed within a tag in the document body and applies only to that tag

9 ☐ **DESIGNING A STYLE RULE—PAGE 1**

- Style sheets are written in CSS (Cascading Style Sheet) language and implement web page formatting and layout
- Language is different from HTML but is the preferred method for formatting web pages

10 ☐ **DESIGNING A STYLE RULE—PAGE 2**

- Format:

selector
{
    property1: *value1*;
    property2: *value2*;
    ...
}

- The *selector* is name of element (e.g. tag or ID or class) or "group" of elements
- The *property* is the name of a format type and the *value* is value assigned to it

11 ☐ **DESIGNING A STYLE RULE—PAGE 3**

- Examples:

```
h1
{
    text-align:center;
}
p
```

```
{
   font-family:Arial;
   color:blue;
}
```

12 **EXTERNAL STYLE SHEETS—PAGE 1**

- A web page can link to a separate (external) style sheet file
- Linked style sheets are created as a *separate document files* with a ".css" extension
- A link tag is placed into the head section of each web document that will implement the style sheet

13 **EXTERNAL STYLE SHEETS—PAGE 2**

- Format:

```
<link href="location/filename" rel="stylesheet" type="text/css" />
```

- Example:

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

15 **EMBEDDED IMPLEMENTATION—PAGE 1**

- Embedded implementation is a method in which the style sheet is embedded into a single web page
- Only the one web document in which it is embedded will be able to use the features in the style list
- Useful if *several elements* in an individual web page have a unique style
- Implemented by placing a style block in the head section

16 **EMBEDDED IMPLEMENTATION—PAGE 2**

- Format:

```
<head>
   ...
   <style type="text/css">
   {
      property1: value1;
      property2: value2;
      ...
   }
   </style>
</head>
```

17 **EMBEDDED IMPLEMENTATION—PAGE 3**

- Example:

```
<head>
   <title>Style Sheet Example</title>
   <style type="text/css">
```

```
        h1
        {
            background-color: rgb(133, 109, 85);
        }
    </style>
</head>
```

18 **INLINE IMPLEMENTATION—PAGE 1**

- Inline implementation places style sheet formatting at any point in the document body
- Involves including a style *attribute* inside the tag; the formatting applies only to that tag *at that location*
- An advantage of this technique is the closing tag for the element *turns off* all the style formatting

19 **INLINE IMPLEMENTATION—PAGE 2**

- Format:

  <*element* style="*property1*: *value1*; *property2*: *value2*; ... ">*content*</*element*>
- Example:

```
<p style="font-family: Arial; color: #FFFF00;">Inline implementation is higher in
    priority</p>
```

20 **STYLE PRECEDENCE AND SPECIFICITY**

- Generally a more specific style is implied instead of a more general one
  - E.g. a style applied to a paragraph is applied before one applied to a section
- For styles implemented at more than one level (inline, embedded, linked) in a website, the order of precedence is:
  - Browser's internal style sheet (install defaults) (highest)
  - User defined styles (user options)
  - Inline
  - Embedded
  - Linked (lowest)

21 **COMMENTS IN STYLE SHEETS**

- Style sheet comments use a form that is similar to *multi-line* comments in C++ and Java
- Format:

```
/*  Comment text  */
```

- Example:

```
/*  Author:  Prof. Carl B. Struck
    The "sa_styles.css" style sheet
*/
```

23 RGB COLORS IN CSS—PAGE 1

- Color values may be specified as RGB values using the rgb function specifying the amounts of red, green and blue as integer values between zero (0) and 255
- There are 16,777,216 color combinations

24 RGB COLORS IN CSS—PAGE 2

- Format:

  colorProperty: rgb(*redN*, *greenN*, *blueN*)
  - Values for *redN*, *greenN and blueN* between zero (0) (none) and 255 (maximum)

25 RGB COLORS IN CSS—PAGE 3

- Example:

```
h1
{
    color: rgb(255, 255, 0);
}
```

  - Renders as yellow (maximum red and green)

26 COLOR NAMES IN CSS—PAGE 1

- There is a list of 140 color names supported by all modern browsers which can substitute for the rgb function
- A partial list of the color names include red, pink, black, gray, yellow, blue, orange, white, beige, fushia, gold, etc.
- Find the complete list at http://www.w3schools.com/colors/colors_names.asp

27 COLOR NAMES IN CSS—PAGE 2

- Format:

  colorProperty: *colorName*;
- Example:

```
body
{
    background-color: antiquewhite;
}
```

28 RGB COLORS WITH HEXADECIMAL NUMBERS—PAGE 1

- Color values may be specified as RGB values with a 6-digit hexadecimal number specifying:
  - The amount of red in the first two digits
  - The amount of green in the middle two digits
  - The amount of blue in the last two digits
- Each of the two-digit hexadecimal numbers are in the range:
  - From zero (00) which is none
  - To 255 (FF) which is maximum

29 **RGB COLORS WITH HEXADECIMAL NUMBERS—PAGE 2**

- Zero (0) through 255 represented as two-digit hexadecimal numbers:
  - 0—00
  - 1—01
  - 2—02
  - 3—03

    ...
  - 253—FD
  - 254—FE
  - 255—FF
    - (Hex digits: A=10, B=11, C=12, D=13, E=14, F=15)
-

30 **RGB COLORS WITH HEXADECIMAL NUMBERS—PAGE 3**

- Format:

  colorProperty: *#hexadecimalNumber*;
  - Values for *redN*, *greenN and blueN* between zero (0) (none) and 255 (maximum)

31 **RGB COLORS WITH HEXADECIMAL NUMBERS—PAGE 4**

- Example:

```
h1
{
   color: #FF00FF;
}
```

  - Renders as magenta (maximum red and blue, no green)

32 **HSL COLORS IN CSS—PAGE 1**

- Color values may be specified as HSL values using the hsl function in which the following values apply:
  - Hue is the measurement in degrees on a color wheel from 0 to 360
    - Zero (0) is red
    - 120 is green
    - 240 is blue

33 **HSL COLORS IN CSS—PAGE 2**

- Color values may be specified as HSL values using the hsl function in which the following values apply (*con.*):
  - Saturation is a percentage measuring color intensity from 0% to 100%
    - 100% is pure color, no shades of gray
    - 50% is 50 percent gray, but can still see the color
    - 0% is completely gray, can no longer see the color

34 **HSL COLORS IN CSS—PAGE 3**

- Color values may be specified as HSL values using the hsl function in which the following values apply (*con.*):
  - Lightness is a percentage measuring how much light is given to the color from 0% to 100%
    - 0% is no light (black)
    - 50% is 50 percent gray light (neither dark nor light)
    - 100% is full lightness

35 **HSL COLORS IN CSS—PAGE 4**

- Format:
  colorProperty: hsl(*hueN*, *saturationN*%, *lightnessN*%)
  - *hueN*, between 0° and 360°
  - *saturationN* between 0% and 100%
  - *lightnessN* between 0% and 100%

36 **HSL COLORS IN CSS—PAGE 5**

- Example:

```
h1
{
   color: hsl(39, 100%, 50%);
}
```

  - Renders as orange (39°, 100% saturation and 50% lightness)

38 **THE "BACKGROUND-COLOR" PROPERTY**

- Formats the *background* color behind an element
- Format:
  background-color: rgb(*redN*, *greenN*, *blueN*) | *color_name*;
- Example:

```
body
{
   background-color: antiquewhite;
}
```

40 **DEPRECATED COLOR ELEMENTS**

- Pre-CSS web pages may use the <font> tag and various attributes (e.g. bgcolor, color, text) to format for color
- These HTML elements are considered deprecated
  - They still work in almost all browsers but are "out-of-date"
- Examples:

```
<body bgcolor="blue" text="255, 255, 255">
<p><font color="green">Welcome to HTML</font></p>
```

41 **THE "RGBA" FUNCTION—PAGE 1**

- The "a" in the rgba function stands for alpha and relates to the amount of transparency in the color in which
- Format:
  rgba(*redN*, *greenN*, *blueN*, *opacity*)
  - The *opacity* is a decimal value between:
    - Zero (0)—0% which is invisible
    - One (1)—100% which is solid

42 **THE "RGBA" FUNCTION—PAGE 2**

- Example:

```
h1
{
   color:rgba(255, 255, 100, 0.7);
}
```

- Opacity in this example is 70%

45 **CONTEXTUAL SELECTORS—PAGE 1**

- Takes advantage of hierarchy of elements so that only elements "descended" (directly or indirectly) from a specific "parent" type are effected
- Format:

```
parent  descendant ...
{
   property1: value1;
   property2: value2;
   ...
}
```

46 **CONTEXTUAL SELECTORS—PAGE 2**

- Example:

```
section h1
{
   color: blue;
}
```

- Only h1 elements in the section are effected

47 **CONTEXTUAL SELECTORS—PAGE 3**

- Contextual selectors take advantage of the rule that the *more specific* style takes precedence over the less specific
- Examples:

```
h1
{
   color: red;
}
```

```
section h1
{
    color: blue;
}
```

- Second example takes precedence over the first

49 **THE "ID" ATTRIBUTE SELECTOR—PAGE 1**

- A style can be applied to an element based on its id
- A hash (#) symbol precedes the id name in the style rule
- Format:

```
#id
    property1: value1;
    property2: value2;
    ...
}
```

50 **THE "ID" ATTRIBUTE SELECTOR—PAGE 2**

- Example:

```
#main
{
    color:blue;
}
```

- Would apply to the following h1 heading:

```
<h1 id="main">My Heading</h1>
```

51 **DEFINING CLASSES**

- The class attribute is a developer-defined name for the tag which can be used to allow styles to be applied selectively
- Format:

<element class="*className*">*content*</*element*>

- Example:
  - <a class="mail" href= ... >
  - Specific anchor tag is defined as having the class name "mail" and, as such, is differentiated from other anchor tags in the same web document

52 **USING CLASS NAME IN A STYLE SHEET—PAGE 1**

- The class name following a dot (.) may used as a selector
- Format:

```
.class
{
    property1: value1;
    property2: value2;
    ...
```

```
}
```

53 USING CLASS NAME IN A STYLE SHEET—PAGE 2

- Example:

```
.mainHeadings
{
    color: blue;
}
```

- Would apply to all elements with the class name "mainHeadings" (even tags of different types but with the same class name)

54 USING CLASS NAME IN A STYLE SHEET—PAGE 3

- The class name may used with a modifier of any selector in stylesheet which allows styles to be applied selectively to an individual tag or tags
- Format:

```
element.class
{
    property1: value1;
    property2: value2;
    ...
}
```

55 USING CLASS NAME IN A STYLE SHEET—PAGE 4

- Example:

```
a.mail
{
    color: blue;
}
```

- Would apply only to a (anchor) tags with the class name "mail"

57 CHOOSING TEXT FONTS—PAGE 1

- The font-family property specifies comma-delimited (comma-separated) list of font face names (known as *typefaces*)
- Format:

  font-family: *fontTypefaces*

  - The browser will select the first "installed" font from *fontTypefaces* list

58 CHOOSING TEXT FONTS—PAGE 2

- Example:

```
p
{
    font-family: 'Arial Black', Arial, sans-serif;
```

```
}
```

- Multi-word font names must be listed in 'single quotes'

59 **GENERIC FONT GROUPS**

- Describes general appearance of a typeface including:
  - serif—small ornamentation at tail end of each character
  - sans-serif—font without ornamentation
  - monospace—each character has same width as all others
  - cursive—mimics handwriting
  - fantasy—highly ornamental; not appropriate for body text

60 **WEB SAFE FONTS**

- Typefaces that display mostly the same in all browsers are considered "web safe" and include:
  - Arial, Arial Black, Century Gothic, Comic Sans MS, Courier New, Georgia, Impact, Lucida Console, Lucida Sans Unicode, Tahoma, Times New Roman (the default in many browsers), Trebuchet MS, Verdana

62 **SETTING FONT SIZE—PAGE 1**

- The font-size property is measured in "length" which can be specified in one of four ways:
  - With a "unit of measure"
  - As a percentage of the size of the containing element
  - With a keyword description
  - With a keyword description that is the size relative to the size of the containing element
- Format:

```
font-size: sizeN | xx-small | x-small | small | medium | large | x-large | xx-large |
           smaller | larger;
```

63 **SETTING FONT SIZE—PAGE 2**

- Example (unit of measure):

```
p
{
   font-size: 2em;
}
```

- Example (percentage relative to size of containing element):

```
p
{
   font-size: 75%;
}
```

64 **SETTING FONT SIZE—PAGE 3**

- Example (keyword description):

```
p
{
   font-size: x-large;
}
```

- Example (keyword description where size is relative to size of containing element):

```
p
{
   font-size: smaller;
}
```

65 CSS UNITS OF MEASURE—PAGE 1

- CSS recognizes several "units of measure" including:
  - % —percentage of containing element
  - in (inch)
  - cm (centimeter)
  - mm (millimeter)
  - em—1 em is equivalent to current "browser" font size; useful since it can adapt automatically to the font that the reader uses
    - Historically "em" equivalent to width of uppercase "M"

66 CSS UNITS OF MEASURE—PAGE 2

- CSS "units of measure" (*con.*):
  - pt (point)—1 pt is the same as 1/72 of an inch
  - pc (pica)—1 pc is the same as 12 points
  - px (pixel)—a dot on the computer screen

68 THE "LETTER-SPACING" PROPERTY—PAGE 1

- The letter-spacing property controls the amount of space between characters (default is no extra space)
- Format:

  letter-spacing: *value*;
  - The value normal is used if an element might be inheriting some other value

69 THE "LETTER-SPACING" PROPERTY—PAGE 2

- Example:

```
h1
{
   letter-spacing: 10px;
}
```

70 THE "WORD-SPACING" PROPERTY—PAGE 1

- The word-spacing property controls the amount of space between words (default is

no extra space)

- Format:

word-spacing: *value*;

71 THE "WORD-SPACING" PROPERTY—PAGE 2

- Example:

```
h1
{
   word-spacing: 10px;
}
```

72 THE "LINE-HEIGHT" PROPERTY—PAGE 1

- The line-height property controls the amount of space between lines of text (default is 100%)
- Format:

line-height: *size*;

73 THE "LINE-HEIGHT" PROPERTY—PAGE 2

- Example:

```
p
{
   line-height: 115%;
}
```

74 CONTROLLING INDENTATION—PAGE 1

- The text-indent will indent the element by specified size
- For a paragraph will indent the first line only
- Format:

text-indent: *size*;

- The value initial is used to return to default if an element might be inheriting some other value
- A negative value is used to create a hanging indent

75 CONTROLLING INDENTATION—PAGE 2

- Example:

```
p
{
   text-indent: 115%;
}
```

77 SETTING THE TEXT'S ITALIC STYLE—PAGE 1

- The font-style property creates *italic* (sloped) text
- Format:

font-style: normal | italic | oblique;

- Generally italic is standard incline while oblique is inclined just a bit

78 **SETTING THE TEXT'S ITALIC STYLE—PAGE 2**

- Example:

```
h1
{
    font-style: italic;
}
```

79 **SETTING THE TEXT'S BOLDFACE PROPERTY—PAGE 1**

- The font-weight property applies boldface to text
- Format:

  font-weight: normal | bold | bolder | lighter | 100 | ... | 900;
  - *Numeric values* between 100 to 900 (in increments of 100's) indicate lighter to heavier weights
    - Some may appear identical if fewer than nine weights installed in browser

80 **SETTING THE TEXT'S BOLDFACE PROPERTY—PAGE 2**

- Example:

```
h1
{
    font-weight: bold;
}
```

81 **SETTING THE TEXT'S LINE PROPERTY—PAGE 1**

- The text-decoration property applies underlining, overlining and ~~strike-through~~ to text
- Format:

  text-decoration: underline | overline | line-through | none;

82 **SETTING THE TEXT'S LINE PROPERTY—PAGE 2**

- Example:

```
h3
{
    text-decoration: line-through;
}
```

83 **"TRANSFORMING" TEXT—PAGE 1**

- The text-transform property applies a specific *text case* to text as in:
  - All in UPPERCASE
  - All in lowercase
  - Just initial Capital letters for each word
- Format:

  text-tranform: capitalize | uppercase | lowercase | none;

84 "TRANSFORMING" TEXT—PAGE 2

- Example:

```
h1
{
    text-transform: capitalize;
}
```

- First letter of each word will be uppercase

85 SETTING THE TEXT'S "SMALL CAPS" STYLE—PAGE 1

- The font-variant property applies SMALL CAPS style to text
- Format:

```
font-variant: small-caps | normal;
```

86 SETTING THE TEXT'S "SMALL CAPS" STYLE—PAGE 2

- Example:

```
ul
{
    font-variant: small-caps;
}
```

87 ALIGNING TEXT HORIZONTALLY—PAGE 1

- The text-align property applies a specific *alignment* to the text of a tag or other element
  - The value justify means an even, not jagged, margin on both left and right sides of an element
- Format:

```
text-align: left | right | center | justify;
```

88 ALIGNING TEXT HORIZONTALLY—PAGE 2

- Example:

```
h1
{
    text-align: center;
}
```

89 ALIGNING TEXT VERTICALLY—PAGE 1

- The vertical-align property applies a specific *vertical alignment* to the text relative to the content around it
- Format:

```
vertical-align: valueN | baseline | sub | super | top | text-top | middle | bottom | text-
    bottom | initial;
```

90 ALIGNING TEXT VERTICALLY—PAGE 2

- Example:

```
h2
{
    vertical-align: 50%;
}
```

- Lowers the element by half of the line height

91 COMBINING FONT FORMATTING IN A SINGLE STYLE—PAGE 1

- The font property applies setting several font values at once including style (italic), variant (small-caps), weight (boldface), size and line height, and typeface (font-family)
- Format:

font: *font-style font-variant font-weight*
font-size[*/line-height*]
font-family1 [, *font-family2* ...];

92 COMBINING FONT FORMATTING IN A SINGLE STYLE—PAGE 2

- Example:

```
p
{
    font: italic small-caps bold 16pt/1.5 Veranda;
}
```

94 SELECTING LIST STYLE TYPES—PAGE 1

- The list-style-type property that applies a bullet or "number" style to unordered and ordered lists
- Styles include a disk (filled-in circle), circle (open circle), square, decimal numbers (with or without leading zeros), Roman numerals in upper or lower case, and upper or lowercase letters in English or Greek

95 SELECTING LIST STYLE TYPES—PAGE 2

- Format:

list-style-type: disk | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-alpha | upper-alpha | lower-greek | upper-greek | none;

96 SELECTING LIST STYLE TYPES—PAGE 3

- Examples:

```
ul
{
    list-style-type: square;
}
ol
{
```

```
    list-style-type: lower-roman;
}
```

101 **PSEUDO-CLASSES—PAGE 1**

- A pseudo-class is the classification of an element based dynamically upon its current position
- Format:

```
selector:pseudo-class
{
    property1: value1;
    property2: value2;
    ...
}
```

102 **PSEUDO-CLASSES—PAGE 2**

- There are five special pseudo-classes that are often applied to hyperlinks (and some other elements) as follows:
  - Links that have not yet visited or clicked:
    a:link { *property*:*value*; ... }
  - Links that have previously been visited:
    a:visited { *property*:*value*; ... }
  - Links that have been clicked:
    a:active { *property*:*value*; ... }

103 **PSEUDO-CLASSES—PAGE 3**

- There are five special pseudo-classes that are often applied to hyperlinks (and some other elements) as follows (*con.*):
  - The mouse is pointing to it (hovering over the hyperlink)
    a:hover {*property*:*value*; ... }
  - The element has received keyboard or mouse focus
    a:focus {*property*:*value*; ... }

104 **PSEUDO-CLASSES—PAGE 4**

- Example:

```
a:hover
{
    color: blue;
}
```

105 **PSEUDO-CLASSES—PAGE 5**

- The hover, active and focus pseudo-classes also can apply to non-hypertext elements
- Example:

```
nav ul li:hover
{
    background-color: blue;
}
```

107 STRUCTURAL PSEUDO-CLASS—PAGE 1

- Items may be classified based on their locations using "structural" pseudo-classes including:
  - :first-of-type—first element that matches specified type
  - :last-of-type—last element that matches specified type
  - :nth-of-type—n^{th} element of specified type
  - :nth-last-of-type—n^{th} from last element of specified type
  - :only-of-type—only element that matches specified type

108 STRUCTURAL PSEUDO-CLASS—PAGE 2

- Format:

```
selector:structural-pseudo-class
{
    property1: value1;
    property2: value2;
    ...
}
```

109 STRUCTURAL PSEUDO-CLASS—PAGE 3

- Example:

```
nav ul li:first-of-type
{
    color: blue;
}
```

110 STRUCTURAL PSEUDO-CLASS—PAGE 4

- The :nth-of-type selector requires an argument that specifies the location of an item in the group *by number*
- Format:

```
selector:nth-of-type(n)
{
    property1: value1;
    property2: value2;
    ...
}
```

111 STRUCTURAL PSEUDO-CLASS—PAGE 5

- Example:

```
nav ul li:nth-of-type(2)
{
    color: blue;
}
```

- Applies blue text to the *second* list item

113 **PSEUDO-ELEMENTS—PAGE 1**

- A pseudo-element is used to apply a style to a specified "part" of an element including:
  - The first letter or line of an element
  - Content to be inserted before or after an element

114 **PSEUDO-ELEMENTS—PAGE 2**

- Format (CSS3):

```
selector::pseudo-element
{
    property1: value1;
    property2: value2;
    ...
}
```

- The pseudo-element may be first-letter, first-line, before or after

115 **PSEUDO-ELEMENTS—PAGE 3**

- Format (prior to CSS3):

```
selector > pseudo-element
{
    property1: value1;
    property2: value2;
    ...
}
```

Advisable to still use this format for backward compatibility with older browsers, e.g. even current version Internet Explorer

116 **PSEUDO-ELEMENTS—PAGE 4**

- Examples:

```
section >  p:first-letter
{
    font-size:200%;
}
section >  p:first-line
{
    text-transform:uppercase;
}
```