3.1 INTRODUCTION TO CSS

What is Cascading Style Sheet (CSS)?

A CSS stylesheet comprises set of rules that are interpreted by the web browser and then applied to the corresponding elements such as paragraphs, headings, etc. in a web document.

The CSS rule consist of statement that defines the style of element in a web page.

- 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

These rules usually follow a specific structure.

3.1 INTRODUCTION

History:

- The CSS1 specification was developed in 1996
  - CSS2 was released in 1998
    - CSS2.1 reflects browser implementations
  - CSS3 are implemented in current browsers

To keep your CSS organized, use CSS comment tags to identify the different sections of CSS within your style sheet.

use C comments (/*...*/)

syntax:

```
/* ... */
```

3.2 LEVELS OF STYLE SHEETS

- **Inline style sheets** appear in the opening tag of html.

- **Document-level / Embedded style sheets** appear in the head section of the HTML document.

- apply to the whole document in which they appear

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

- can be applied to any number of documents.

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

Note:

- 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)

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

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

- External style sheets can be validated

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

3.2 LEVELS OF STYLE SHEETS – (CONTINUE)

1. INLINE STYLE SHEET

The style specifications are placed within the html elements.

It is the most common method of attaching a style sheet to an HTML document.

The inline style sheet is used to apply declaration style to an individual element in a particular document.

Note:

Inline style sheet should be avoided in two cases:

1. If we want to apply the same style declaration to different elements every time.
2. Inline style sheet mixes the content with the presentation. So, if you want to avoid this mixing up, don't use Inline style sheet.

3.3 STYLE SPECIFICATION FORMATS

1. INLINE CSS

Style sheet appears as the value of the style attribute

- **General Syntax:**

```
<tag     style = "property_1: value_1;
                  property_2: value_2;
                   …
                  property_n: value_n“>
…..</tag>
```

```
<html>
<head>
<title> Inline CSS example</title>
<meta http-equiv="content-style-type"
content="text/css">
</head>
<body style="background:orange">
<h1 style="color:White; font-
family:arial; font-size:14pt; text-
transform:uppercase; text-align:left;">
This is an example of inline css </h1>
</body>
```

3.3 STYLE SPECIFICATION FORMATS (CONTINUED)

2. EMBEDDED CSS

- The <style> tag must include the type attribute,

set to "text/css"

- General form/Syntax:

<style type = "text/css">

rule list

</style>

```
<!DOCTYPE html>
<html>
    <head>
        <title> Embedded CSS</title>
        <style type="text/CSS">
            body {
                    background-color:#ccffff;
                }
            h1 { color: purple; font-family: arial; font-size: 30
px; text-transform: uppercase; text-align: left;}
        </style>
    </head>
    <body>
        <h1> This is an example of embedded CSS</h1>
        <h1> B E </h1>
    </body>
</html>
```

Output

3.3 STYLE SPECIFICATION FORMATS (CONTINUED)

2. EMBEDDED CSS EXAMPLE

```
<html>
<style type="text/css">
<!--
  h4 {
  font: 17pt  Arial,Helvetica";
  font-weight: bold;
  color: maroon
}

  h2 {font: 15pt  "Courier"; font-weight: bold;  color: blue}
  p  {font: 12pt  "Arial,Helvetica"; color: black}
-->
</style>
<body>
...
</body>
</html>
```

3.3 STYLE SPECIFICATION FORMATS (CONTINUED)

3. EXTERNAL CSS

The CSS files are kept separately from an HTML document.

External CSS file contains only CSS code and it is saved with a “.css” extension.

Use a <LINK> tag instead of <STYLE> tag in the <HEAD> section of the HTML document.

Advantages:

1. It is a “true separation” of style and content.
2. It is easier to reuse CSS code in any separate file.
3. It is easier to edit the css rule.

3. Demonstration of external style sheet. (you will have two or more documents

Save the following program as external.css

```
body { background:#ccffff;}
h2,p {
       color: green;
       font-family:arial;
       text-align:center;
       }
p i{
     color: orange;
     border-style: solid;
     font-weight: lighter;
     }
.ex{color:purple}
```

Save the following program as external.html and link above CSS file in it.

```
<html>
    <head><title>Extenal style sheet</title>
        <link red= "stylesheet" type= "text/CSS" href="external.css">
    </head>
    <body>
        <h2> It is an example of External style sheet</h2>
        <p class="ex"> This is a "true separation" of style and content</p>
        <p><i> External CSS </i> </p>
    </body>
</html>
```

IMPORTANT TAGS

- **<STYLE> tag**
  Defines document level styles

```
<STYLE TYPE="text/css">      /* Styles go here... */    </STYLE>
```

- **<LINK> tag**
  References external style sheets. Allows for alternates.

```
<LINK REL="stylesheet" HREF="default.css" TYPE="text/css">
```

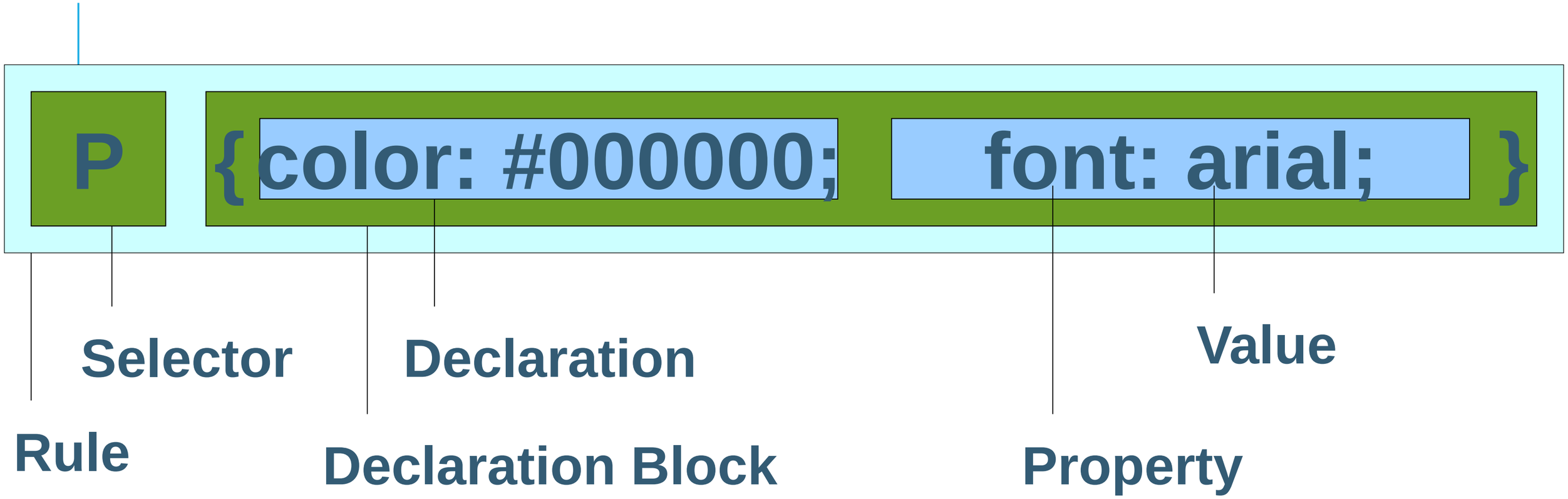
- **STYLE attribute**
  Defines inline styles for a specific block of HTML code

```
<P STYLE="color: #FF0000; font-weight: bold;"> some text </P>
```

CSS: SYNTAX TERMINOLOGIES

- **@import Directive**
  Loads an external style sheet. Not supported in some older browsers.
- **Rules**
  Defines which styles to apply to which elements
- **Selectors**
  Specifies the element or type of element that style affects
- **Declarations**
  Specifies CSS properties and values

CSS: RULE STRUCTURE



CSS: SELECTOR TYPES

```
P { color: black; }                                  /* Element Selector */
P, H1, H2 { color: black; }                          /* Grouping Selector */
* { color: black; }                                  /* Universal Selector */
P.urgent, .Error { color: black; }                   /* Class Selector */
#Menu { color: black; }                              /* ID Selector */
*[title], A[href][title] { color: black; }           /* Attribute Selector */
A[title="home page"] { color: black; }               /* Exact Attribute Selector */
A[title="foo"] { color: black; }                     /* Partial Attribute Selector */
*[lang|="en"] { color: black; }                      /* Particular Attribute Selector */
UL  LI  UL { color: black; }                         /* Descendant Selector */
P > STRONG { color: black; }                         /* Child Selector */
H1 + P { color: black; }                             /* Adjacent Sibling Selector */
A:hover { color: black; }                            /* Pseudo-Class Selector */
P:first-letter { font-size: 200%; }                  /* Pseudo-Element Selector */
```

CSS: COMMON DECLARATION PROPERTIES

background
background-attachment
background-color
background-repeat
border
bottom
color
cursor
display
float
font
font-family
font-size
font-style
font-weight
height
left

letter-spacing
line-height
list-style-image
list-style-position
list-style-type
margin
overflow
padding
position
right
text-align
text-decoration
text-indent
text-transform
top
vertical-align
visibility

white-space
width
word-spacing
word-wrap
z-index

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>
```

3.4 Selector Forms

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
  - 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

PSEUDO-ELEMENTS AND PSEUDO-CLASSES

There are two types of keywords that you can combine with selectors: Pseudo-elements and Pseudo-classes

1. Pseudo-classes

A pseudo-class is used to define a special state of an element.

It can be used to:

Style an element when a user mouses over it

Style visited and unvisited links differently

Style an element when it gets focus

Syntax:

```
selector:pseudo-class {
  property: value;
}
```

Example:

```
/* unvisited link */
a:link {
  color: #FF0000;
}
/* visited link */
a:visited {
  color: #00FF00;
}

/* mouse over link */
a:hover {
  color: #FF00FF;
}
/* selected link */
a:active {
  color: #0000FF;
}
```

PSEUDO-ELEMENTS AND PSEUDO-CLASSES-CONT.

There are two types of keywords that you can combine with selectors: Pseudo-elements and Pseudo-classes

2. Pseudo-elements

A CSS pseudo-element is used to style specified parts of an element.

Example, it can be used to:
Style the first letter, or line, of an element
Insert content before, or after, the content of an element.

Syntax:

```
selector::pseudo-element {
  property: value;
}
```

```
h1:first-letter { font-size: 200%; color:red;}
p:first-line { color: pink;}
```

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 overridden

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)

- 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*

→ SHOW `text_space.html` and display

3.7 *List properties*

- `list-style-type`

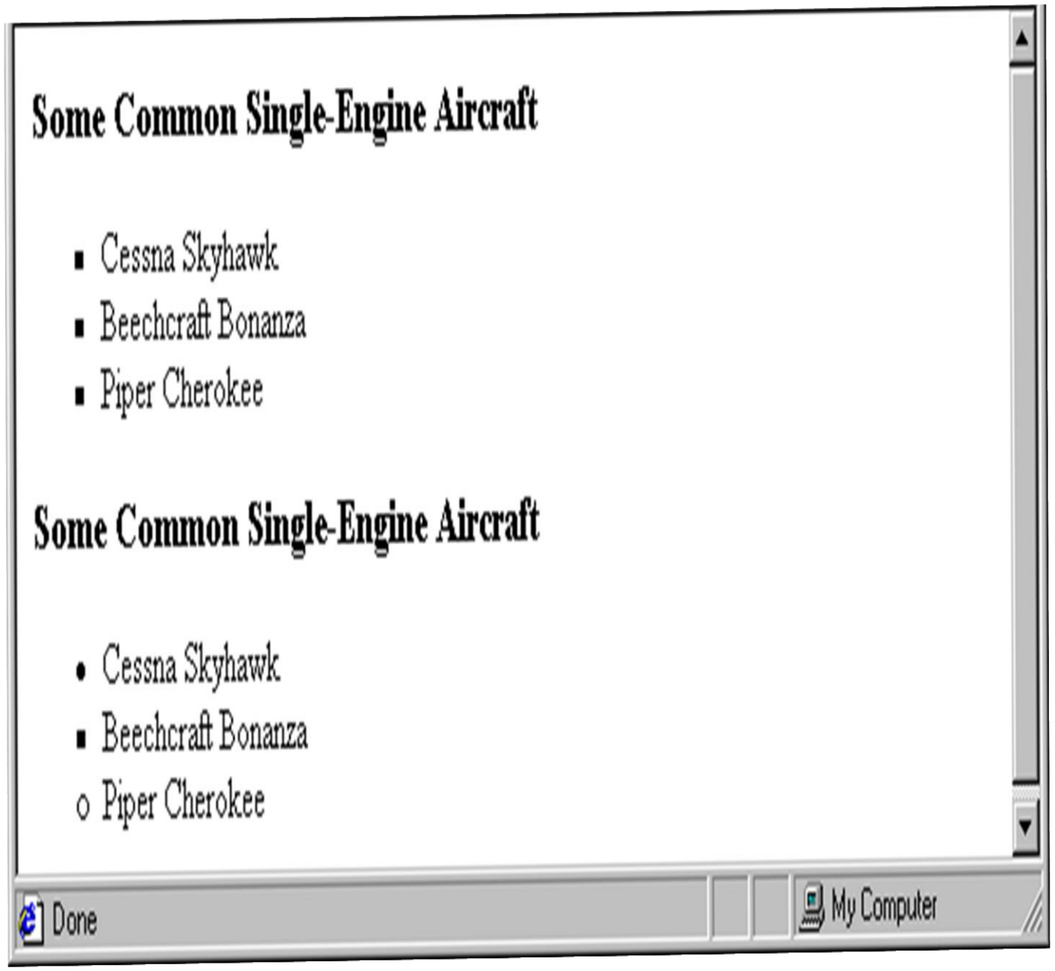
- *Unordered lists*
  - Bullet can be a disc (default), a square, or a circle

  - 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

MODIFYING LIST ELEMENTS

In HTML, by default, unordered lists (`<ul>`) appear as bullets and ordered lists (`<ol>`) appear as numbers.

Using CSS, you can modify how list items appear (Internet Explorer only recognizes the *italicized* values):

- Properties:
  `list-style`, `list-style-type`, `list-style-image`, `list-style-position`
- Values:
  `disc`, *`circle`*, *`square`*, *`decimal`*, `decimal-leading-zero`, *`lower-roman`*, *`upper-roman`*, *`lower-alpha`*, *`upper-alpha`*, `lower-greek`, `lower-latin`, `upper-latin`, `hebrew`, `armenian`, `georgian`, `cjk-ideographic`, `hiragana`, `katakana`, `hiragana-iroha`, `katakana-iroha`, *`none`*, *`url("url-of-graphic.gif")`*, *`inside`*, *`outside`*

Examples:

```
ul { list-style: disc; }
ol { list-style: upper-roman;}
li { list-style: url("http://www.foo.com/images/blackball.gif");}
ul li { list-style-position: inside;}
```

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 display 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`

This is a picture of a Cessna 210. The 210 is the flagship single-engine Cessna aircraft. Although the 210 began as a four-place aircraft, it soon acquired a third row of seats, stretching it to a six-place plane. The 210 is classified as a high performance airplane, which means its landing gear is retractable and its engine has more than 200 horsepower. In its first model year, which was 1960, the 210 was powered by a 260 horsepower fuel-injected six-cylinder engine that displaced 471 cubic inches. The 210 is the fastest single-engine airplane ever built by Cessna.

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`

Color Specification

“color: yellow” (can be black, white, red, blue, …)

“color: rgb(255, 255, 0)”

“color: #FFFF00”

For “Web-safe colors”,
Only use hex values:

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

FONT AND TEXT STYLING

When choosing a font, there are several things to keep in mind:

1. Not everyone has the same set of fonts.
2. If you use a font that the visitor doesn't have, the page will display in the default font (usually Times), unless you provide more choices. To do this, add more than one font in your declaration, and always end with the font family (serif, sans-serif, or monospace):

```
font-family: Verdana, Arial, Helvetica, sans-serif
```

3. Documents designed to be printed tend to look better in Serif fonts (Times, Georgia, Book Antiqua, etc.)
4. Documents designed to be viewed onscreen tend to look better in Sans-serif fonts (Verdana, Arial, Helvetica, etc.)

To apply a font to the entire web page, modify the body tag:

```
body {font-family: Verdana;}
```

To apply a font to a specific section of text, create a class, and use the span tag with that class:

```
.neatstuff {font-family: Comic Sans MS;}

<span class="neatstuff">This is in Comic Sans</span>
```

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)