Laboratory Activity 4

CSS

Objectives:

At the end of this activity, the students are expected to:

1. Learn how to use CSS with HTML
2. Be familiar with different types of styles
3. Develop a simple website integrating CSS

***(Notes are adapted from** http://www.ntu.edu.sg/home/ehchua/programming/index.html#Web)*

1. Introduction to CSS

1.1 What Style Sheet?

A *Style Sheet* is *a collection of style rules* that can be applied to *a selected set of HTML elements*. A style rule is used to control the appearance of HTML elements such as its font properties (e.g., type face, size, weight), color properties (e.g., background and foreground colors), alignment, margin, border, padding, and positioning. This is the same as the style in any publishing software like WinWord or LaTex.

The word "cascading" means that multiple style rules can be applied to the same HTML element simultaneously. The browser follows a certain "cascading order" in finalizing a style to format an HTML element in a predictable fashion.

1.2 A Sample CSS Style Sheet

```
/* A CSS Style Sheet - This is a comment */
body {
  font-family: "Segoe UI", Tahoma, Helvetica, Arial, Verdana, sans-serif;;
  font-size: 14px;  /* in pixels (px) */
  margin: 10px, 0, 10px, 0;  /* top right bottom left */
  padding: 0
}
h1, h2, h3, h4, h5, h6 {
  font-family: "Trebuchet MS", "Segoe UI", Helvetica, Tahoma, Arial, Verdana, sans-serif;
  color: red;
  background: black;
  font-style: italic;
  font-weight: bold;
  text-align: center;
}
p {
  text-align: justify;
  font-size: 14px;
  color: #000000;  /* black */
}
```

Example of CSS and HTML usage:

index.html

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="styles.css" />
</head>
<body>
<h1 >First Page </h1>

<h3>
HI! My name is LBYCPG3. Today, I have learned about HTML and CSS.
Lets talk about CSS…..
</h3>

</body>
</html>
```

styles.css

```
h1 { color: red}
h3 { color: white}
body { background-color:blue }
```

1.3 CSS Syntax

CSS is a language by itself. It has its own syntax, which is totally different from HTML! (How many syntaxes you have to know to program the web?!).

A CSS style sheet is a *collection of style rules*:

1. A *style rule* consists of a *selector* which selects the HTML elements it operates upon (e.g., <body>, <p>, <ul>), and a list of style property names and values enclosed in braces { }, as follows:

```
selector {
  property-name-1: property-value-1-1, property-value-1-2, ... ;
  property-name-2: property-value-2-1, property-value-2-2, ... ;
  ......
}
```

For example,

```
body {   /* Apply to <body> and possibly its descendants */
  font-family: "Segoe UI", Tahoma, Helvetica, Arial, Verdana, sans-serif;;
  font-size: 14px;
  margin: 10px, auto, 10px, auto;  /* top right bottom left */
  padding: 0;
}
```

The *selector* selects the <body> tag. Hence, the defined style is applied to the <body>...</body> element. Many (but not all) of the CSS properties (such as color, font) are *inherited* by the descendents, unless it is overridden by another style definition.

The property *name* and *value* are separated by a colon ':' in the form of *name*:*value*. Multiple values are separated by commas ',' or whitespace. Values containing space must be quoted, e.g., "Times New Roman" or 'Times New Roman'.

The property *name*:*value* pairs are separated by semicolon ';'. You can omit the last semicolon ';' before the closing brace '}'. But I recommend that you keep it, so that it is easier to include new entries without a missing ';'.

Extra whitespaces (blank, tab and newline) are ignored.

If the same styles are applicable to more than one tags, the selectors can be grouped together in one single rule, separated by commas ','. For example, the following rule apply to HTML tags <h1> to <h6>:

```
h1, h2, h3, h4, h5, h6 { text-align: center; font-family: "Trebuchet MS", "Segoe UI",
Helvetica, Tahoma, Arial, Verdana, sans-serif; }
```

Comments can be inserted inside the sheet sheet enclosed between /* and */. (The end-of-line comment // does not work in CSS?!)

Take note that CSS and HTML have different syntaxes. In HTML, tags' attributes uses '=' to separate the *name* and *value* pair, in the form of *name*="*value*"; attributes are separated by spaces. For example,

```
<img src="logo.gif" height="10" width="20">
```

1.4 Types of Styles

There are three places where you can define style rules:

1. **Inline Style**: included inside a particular HTML tag's attribute style="*style-rules*", and is applicable to that particular HTML element only.
2. **Embedded Style Sheet**: embedded inside the <style>...</style> tag in the HEAD section of the HTML document. The styles are applicable to that document only.
3. **External Style Sheet (Recommended)**: stored in an external file, which is then linked to HTML documents, via a <link> tag in the HEAD section. An external style sheet can be applied to many HTML documents to ensure uniformity (e.g., all pages in your website).

Inline Styles
To apply inline style to an HTML element, include the list of style properties in the style attribute of the tag.
For example,

```
<!DOCTYPE html>
<html>
<body>
  <p style="font-size:18px; font-family:cursive">This paragraph uses 18px cursive font.</p>
  <p>This paragraph uses default font.</p>
  <p>This paragraph uses <span style="font-size:20px">20px inside this span</span>
but default font size here.</p>
</body>
</html>
```

This paragraph uses 18px cursive font.

This paragraph uses default font.

This paragraph uses 20px inside this span but default font size here.

The scope of an inline style is limited to that particular tag. Inline style defeats the stated goal of style sheets, which is to separate the document's content and presentation. Hence, inline style should be avoided and only be used sparsely for *touching up a document*, e.g., setting the column width of a particular table.

Embedded Styles

Embedded styles are defined within the <style>...</style> tag in the HEAD section. For example,

```
<!DOCTYPE html>
<html>
<head>
  <style type="text/css">
body { background-color:cyan }
h2 { color:white; background-color:black }
p.cursive { font-size:18px; font-family:cursive }
    p.f20px   { font-size:20px }
</style>
</head>
<body>
  <h2>H2 is white on black</h2>
  <p>This paragraph is normal.</p>
  <p class="cursive">This paragraph uses 18-px cursive font.</p>
  <p class="f20px">This paragraph uses 20-px font.</p>
</body>
</html>
```

- The scope of the embedded styles is the current HTML document.

- Embedded styles separate the presentation and content (in the HEAD and BODY sections) and can be used if page-to-page uniformity is not required. That is, this set of styles is used for only one single page!?
  (I use embedded style in this article for illustration, but you should use external style in production.)
- The attribute type="text/css" in the <style> opening tag is not needed in HTML 5.

External Style Sheets

External styles are defined in a external file, and referenced in an HTML document via the <link> tag in the HEAD section.

For example, we define these style rules in a file called "testExternal.css":

```
/* testExternal.css */
body    { background-color:cyan; color:red; }
h2      { background-color:black; color:white; text-align:center; }
p       { font-size:12pt; font-variant:small-caps; }
p.f24pt { font-style:italic; font-size:24pt; text-indent:1cm; }
#green  { color:green; }
```

This HTML document references the external style sheet via the <link> tag in the HEAD section:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="testExternal.css">
</head>
<body>
  <h2>H2 is white on black</h2>
  <h2 id="green">This H2 is green on black</h2>
  <p>The default paragraph uses 12-pt small-cap font.</p>
  <p class="f24pt">This paragraph uses 24-pt, italics font with text-indent of 1cm.
  It inherits the small-cap property from the default paragraph selector.</p>
</body>
</html>
```

The main advantage of external style sheet is that the same styles can be applied to many HTML pages to ensure *uniformity* in presentation for your entire website. External style sheet is the now-preferred approach.

Note: HTML 5 does not requires the type="text/css" attribute in <link> tag.

1.5 Using CSS for Styling

To use CSS to style your website and to get the full benefit of CSS, you need to properly structure your web pages.

HTML Division and Span Tags

Two HTML tags, division ... and span ... are primarily designed for applying CSS styles. They can be used to create *partitions* in an HTML document to represent logical sections (such as header, content, footer). is a*block element* which is rectangular in shape; while is an *inline element* which spans a sequence of characters.

Modern-day HTML pages use and extensively to layout the document via CSS. (Older HTML pages uses tables and frames, which should be avoided. HTML 5 introduces new tags such as <header>, <footer>, <section>, <nav>, <article>to help you better organize your page.)

You can treat and as *generic* tags for identifying contents, which shall be further qualified via the id or class attribute. Similarly, and does not have any inherit visual properties (unlike <h1>), you need to attach presentation properties via CSS.

First of all, partition your web page in *logical sections* via ... , such as header, content, navigation menu, footer, etc.

CSS Selector

As mentioned earlier, a CSS *selector* is used to select a group of HTML elements to apply the defined styles. The selection can be carried out via:

1. HTML tag name, e.g., body, p, ol, table, and div.
2. HTML tag's id attribute, e.g., .
3. HTML tag's class attribute, e.g., .
4. others.

HTML Tag Attributes id="*idValue*" and class="*classname*"

All the HTML tags supports two optional attributes: id="*idValue*" and class="*className*".

1. You can assign an id="*idValue*" to an HTML element to uniquely identify that element. The id value must be unique within the HTML document. In other words, no two elements can have the same id value. The id attribute is used by CSS as well as JavaScript to select the element. For example,

```
<div id="header"><h1>Header Section</h1></div>
<div id="content"><h1>Content Section</h1></div>
<div id="footer"><h1>Footer Section</h1></div>
```

2. The class value needs not be unique. That is, the same class value can be assigned to many HTML elements. In other words, these HTML elements form a *sub-class*. The class attribute is primarily used by CSS to apply a common set of styles to all the elements of the same class. (HTML 5 can also selects elements based on class name). For example,

```
<p class="highlight">A highlighted paragraph</p>
```

3. A class may contain multiple values, separated by space, e.g.,

```
<p class="highlight underline">This paragraph element has two class values</p>
```

Another highlighted paragraph

Partition Your Web Page into Logical Sections

The id and class attributes are important for to identify partitions created via and , for applying styles.

The first step in web page design is to partition your web page in logical sections using the ... tag, and assign id or class to each of the divisions. Use id if the division is unique (in formatting); otherwise, use class (more than one divisions have the same formatting).

Example

The following HTML page is divided into three divisions, with unique id's of "header", "content" and "footer" respectively. Selected texts are marked with the tags, with class of "green" and "blue".

```
<html>
<head>
<style>
  /* style-definitions - see below */
</style>
</head>

<body>
<div id="header">
  <h1>Heading</h1>
</div>

<div id="content">
<h2>Hello</h2>
<p>Lorem ipsum dolor sit amet, <span class="green">consectetur adipisicing</span> elit, sed do
eiusmod <span class="green">tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.
Duis aute irure dolor in</span> reprehenderit in voluptate <span class="blue">velit
esse</span>
cillum dolore eu fugiat nulla pariatur.</p>
</div>

<div id="footer">
  <p>Footer</p>
</div>
</body>
</html>
```

CSS Tag-name, Id and Class Selectors Tag-name Selector (or Tag Selector or Element Selector): A CSS *tag-name selector* selects HTML elements based on the *tag-name*, e.g.,

```
/* tag-name selector */
h2 { background-color:black; color:white; text-align:center; }
```

ID-Selector: You can use CSS to set the display style for each of the three divisions, via the so-called *idselector*. A CSS *id-selector* begins with a '#' followed by the id's value. It selects a specific element. For example,

```
/* id-selector (id value must be unique) */
#header   { font-size:18px; color:cyan; }
#content  { font-size:14px; color:black; }
#footer   { font-size:12px; color:orange; }
```

Class Selector: CSS also provides the so-called *class-selector* for selecting elements of the same class value.
A CSS *class-selector* begins with a '.' followed by the classname. For example,

```
/* class-selector (class value needs not be unique) */
.green { color:green; text-decoration:underline; }
.blue  { color:blue;  }
```

Note: Because of the wide popularity of using 's with id="header", id="footer" to organize an HTML page, HTML 5 defines new tags such as <header>, <footer>, <section>, <article>, <nav> to help you to better organize the HTML page.

1.6 Tag's title Attribute

Besides the id and class attributes, you can also assign an attribute title="*label*" to an HTML tag. The title's label shows up as *tool tip* when you points your mouse pointer at the element.

1.7 Types of CSS Selectors

As illustrated in the previous example, a CSS selector can select a set of HTML elements based on (a) tag name,
(b) tag's id attribute, (c) tag's class attribute, and (d) their *many many* combinations. (Read "The 30 CSS
Selectors you Must Memorize". CSS is really humongous!?)

I shall list the frequently-used types of selectors here.

T - Tag-Name Selector (aka Tag Selector, or Type Selector, or Element Selector)
A *tag-name selector* selects all elements of the given tag-name. The syntax is:

tag-name { *style-definitions* }

Example:

```
h2 { background-color:black; color:white; }
```

S1, S2 - Group Selector

You can apply the same style definitions to multiple selectors, by separating the selectors with a commas ','. The syntax is,

selector-1, *selector-2*, ... { *style-definitions* }

Example:

```
h1, h2, h3 { background-color:black; color:white; }
```

T1 T2 - Descendant Selector

You can define a style rule that takes effect only when a tag occurs within a certain *contextual* structure, e.g., descendent, child, first-child, sibling, etc.

To create a *descendant selector*, list the tags in their hierarchical order, with no commas separating them

(commas are meant for grouping selectors). For example,

```
ul li { color:red; } ul ul li {
color:blue; } ul ul ul li {
color:green; }
```

The first-level list items are in red; second-level in blue; and third-level in green.

Note: In T1 T2 { ... }, T2 is a descendant of T1 regardless of the generation.

T1 > T2 - Child Selector

A contextual selector where T2 is an immediate child of T1.

T1:T2 - First-Child Selector

A contextual selector where T2 is the *first* child of T1.

T1 + T2 - Adjacent Selector (aka Sibling Selector)

The style is applied to tag T2, only if it follows immediately after tag T1. T1 and T2 are siblings in the same hierarchical level.

.C - Generic Class Selector

The *generic class selector*, which begins with a dot '.' followed by the classname, selects all elements with the given classname, regardless of the tag name. For example,

```
.f14px_i { font-size:14px; font-style:italic; }
.f16px_b { font-size:16px; font-weight:bold; }
.red { color:red; }
.underline { text-decoration:underline; }
<p class="f14px_i">Text is 14px and italic.</p>
<p class="f16px_b">Text is 16px and bold.</p>
<p class="red">Text is in red.</p>
<h5 class="red">Text is in red.</p>
```

Take note that the class attribute may contain multiple values. This means that you can apply multiple class style rules to an HTML tag. For example,

```
<p class="f14px_i underline">Text is 14px and italic, and underlined.</p>
<p class="f16px_b red underline">Text is 16px and bold, in red and underlined.</p>
```

T.C - Class Selector

The selector T.C selects all tag-name T with classname of C. This is a restricted form of the generic class selector, which applies to the specific tag-name only.

An HTML tag (such as <p>) can be sub-divided into different *style sub-classes* via the class attribute. This subclass mechanism allows us to apply different styles to different subclass of a particular tag.

For example,

```
p         { color:black; }   /* default style for all <p> tags */
p.red     { color:red;   }   /* applicable to <p class="red"> tags (override default) */
p.blue    { color:blue;  }   /* applicable to <p class="blue"> tags (override default) */ h1, h2,
h3 { color:green; }   /* default style for <h1>, <h2> and <h3> tags */ h3.white   { color:white;
}   /* applicable to <h3 class="white"> tags (override default) */ h3.upper   { text-
transform:uppercase; }
```

```
<p>This paragraph is in black (default style)</p>
<p class="red">This paragraph, of class="red", is in red.</p>
<p class="blue">This paragraph, of class="blue", is in blue.</p>
<h2>H2 in green (default style)</h2>
<h3>H3 in green (default style)</h3>
<h3 class="white upper">This H3, of class="white", is in white</h3>
```

Note: Do NOT start a class-name with a number! (This is the same restriction for identifiers in most of the programming languages.)

#D - ID Selector

The id-selector, begins with a '#' followed by the id's value, select a specific element with the given unique id value (or none). Recall that the id value must be unique in an HTML document.

You can use 's with unique id to divide the document into parts of different styles. For example,

```
/* ID selector for the 3 major division of the document */
#header     { font-size:16px; align:center; font-style:bold; }
#header h1  { text-transform:uppercase; }                 /* contextual selector */
#content    { font-size:14px; align:justify; }
#content h3 { color:#FF0000; text-decoration:underline; } /* red, underline */
#footer     { font-size:12px; align:right; }
#footer p   { color:#00FF00; text-decoration:none; }      /* green, not underline */
```

```
<body>

<div id="header">
  <h1>H1 in the "header" division</h1>
  <h3>H3 in the "header" division</h3>
  <p>Paragraph in "header" division</p>
</div>
<div id="content">
  <h1>H1 in the "content" division</h1>
  <h3>H3 in the "content" division</h3>
  <p>Paragraph in "content" division</p>
</div>
<div id="footer">
  <p>Paragraph in "footer" division</p>
</div>
</body>
```

T#D - Tag's ID Selector

Same as above but only if the id is defined under the tag. Since id value is supposed to be unique, this serves more as proper documentation.

* - selector (Universal Selector)

The * universal selector selects ALL the tags in the document. For example,

```
* { margin:0; padding:0; }  /* all tags have margin and padding of 0 */
```

a:link|visited|focus|hover|active - Pseudo-Class Selector

CSS pre-defines a number of pseudo-classes for anchor tag <a>, namely, a:link (unvisited link), a:visited (visited link), a:focus (on focus via tab key), a:hover (mouse pointer hovers over), a:active (clicking the link). Note that colon ":" is used to identify pseudo classes instead of "." for ordinary classname. For example,

```
a       { font-size:14px; }  /* all <a> tags */ a:link    { color:red; }      /* unvisited link */ a:visited { color:green; } /* visited link */ a:focus   { color:lightblue; } /* on focus via tab key */ a:hover   { color:blue; }     /* mouse over link */ a:active  { color:black; }    /* currently selected link */
```

Note:

- The order is important for anchor pseudo-classes in applying styles. It shall be link-visited-focushover-active (LVFHA).
- It is called pseudo-class, as it sub-divide the <a> tag into four sub-classes that cannot be marked out manually.
- You can further restrict the selection via a.*classname*:*pesudo-class*.

Another Example,

```
a {   font-size:10pt;   font-
decoration:underline;
color:red;
}
a.blue:link  { color:blue;  }
a.green:link { color:green; }
a:hover {   font-
decoration:none;
color:yellow;
}
```

```
<a class="green" href="http://www.aaa.com">www.aaa.com</a>
<a class="blue" href="http://www.bbb.com">www.bbb.com</a>
<a href="http://www.ccc.com">www.ccc.com</a>
```

The anchor pseudo classes can be combined with id-selectors (as a descendant selector), so that the appearance of the links is different for the different divisions of the document, e.g.,

```
a  { color:black }  /* default for a:link, a:visited, a:active */
#header a:hover   { color:blue  }
#footer a:hover   { color:green }
```

These pseudo classes is often used with <a> tag. But some of them, such as :hover, :focus, :active can also be applied to other elements, e.g., <p>, <li>, etc.

T:first-line and T:first-letter Pseudo-Elements Selector

The selector p:first-line and p:first-letter select the first line and the first letter of the <p> elements. They are called pseudo-elements because they select a portion of an element that is hard to be marked out (e.g., first line). They can be applied to many tags such as <p>, , , <li> and etc.

You can restrict the selection by applying id or class. For example, p.intro:firstline, em.special:first-letter.

1.8 Inheritance

Many (but not all) CSS properties, such as color properties, affect not only the elements selected by the selector, but also inherited by their descendants. Some properties (such as border, margin,

padding, width and height) are not inherited. You can use a special property value called "inherit" to force the inheritance. For example,

```
<!DOCTYPE html>
<html>
<head>
<style>
p { border: 5px solid red; }
.inherit-border { border: inherit; }
</style>
</head>
<body>
<p>The <em>border</em> property is not inherited.</p>
<p>The <em class="inherit-border">border</em> property is inherited.</p>
</body>
</html>
```

Although is nested under the <p> tag, the border property is not inherited from the parent element. That is, you will not see a border around the 's content. We can force the inheritance by assigning a special value "inherit" as shown.

1.9 Resolving Conflicting Rules: Specificity and Location

The Law of Specificity states that "the more specific the selector, the stronger the rule". For example,

```
<!DOCTYPE html>
<html>
<head>
<style>
p     { color:black; }
p.red { color:red;   }
p#id1 { color:yellow; background:lightblue; }
p#id2 { color:blue;  }
p#id1 { color:green; }  /* Override the color property in the earlier rule */
</style>
</head>
<body>
<p id="id1">Paragraph with "id1" - green</p>
<p id="id2">Paragraph with "id2" - blue</p>
<p class="red">Paragraph of class "red" - red</p>
<p id="id1" class="red">Paragraph with "id1" and class "red" - green</p>
<p>Paragraph without id and class</p>
</body>
</html>
```

The p selector is the most general, which selects all the <p> elements; the p.red selects a group of

<p> elements with attribute class="red"; the p#id1 and p#id2 selects only one element each with the unique id value. The id-selector is the most specific (and takes precedence); followed by class-selector; and followed by general tag-name selector.

If the "Law of Specificity" cannot resolve the conflict, apply the "Law of Locality". The style-rule that read in last by the browser takes effect. In the above example, there are two selector for id1, the later overrides the color property defined earlier.

The inline style (applied to a specific tag) overrides the embedded style and external style sheet.

For embedded style and external style sheet, the order of <link> (for external style sheets) and <style> (for embedded style) tags in the <head> section determine the precedence. You may place <link> before or after <style> and there could be multiple <link> tags for multiple external style sheets. Again, the last rule that read in by the browser takes effect.

You can override all the cascading rules by appending a special value "!important", e.g.,

```
p { color:blue !important; background-color:grey; }  /* color cannot be overridden */ p
{ color:red; background-color:lightblue; }       /* override background-color */ <p>color
is blue but background is lightblue</p>
```

1.10 Validating CSS Files

You can use online services such as http://jigsaw.w3.org/css-validator/ to validate your CSS file.

1.11 Style Properties

For a complete list of the style properties, you may refer to the CSS specification (@ W3C), or online reference (@ http://www.w3schools.com/css/css_reference.asp).

1.12 Color Properties

Specifying Color

Color can be expressed as:

1. RGB hexadecimal triplets in the form of *#rrggbb*, where *rr*, *gg*, *bb* are values of red, green and blue. The values are between 00 and FF, in hexadecimal. For example, #12ABFF. The color value #112233 can be shorthand as #123.
2. RGB decimal values in the form rgb(*r*, *g*, *b*), where *r*, *g*, *b* are between 0 and 255, in decimal.
3. RGB percentages in the form of rgb(*r*%, *g*%, *b*%), where values between 0% and 100%. For example, rgb(0%, 50%, 100%) is the same as rgb(0, 128, 255) and #0080FF.
4. RGBA decimal values in the form rgba(*r*, *g*, *b*, *a*), where *r*, *g*, *b* are between 0 and 255 (decimal), and *a* is between 0 and 1. The *a* (alpha channel) is used to control the transparency/opacity, with a=1 for opaque; and a=0 for totally transparent.

5. The 16 pre-defined English color names as follows. These 16 colors are numerically generated and are really really ugly. You should avoid using them!! Many browsers also support other color names such as lightblue, lightgreen, etc.

CSS's Color Properties

The most important color properties are color and background-color: color: *#rrggbb*|rgb(*r*,*g*,*b*)|rgba(*r*,*g*,*b*,*a*)|*color-name*

Set the color of the text (or foreground).

background-color: *#rrggbb*|rgb(*rrr*,*ggg*,*bbb*)|**rgba(*r*,*g*,*b*,*a*)**|*color-name*|transparent

Set the background color of an element. The default is transparent.

Color values are inherited by descendants.

1.13 Length Measurements

Many CSS properties, such as width, height, margin, border, padding, font-size and line-height, require a length measurement. For example,

```
p {
  width: 80%;    /* 80% of the parent's width */   margin: 10px;
/* pixels */   border: 5mm;   /* millimeters */   padding: 0;
font-size: 1.2em;  /* 1.2 times of the parent's font-size */   line-
height: 1.5;  /* 1.5 times of the current font-size */ }
```

There are two types of length measurements: *relative* (to another length properties) and *absolute* (e.g., inches, centimeters, millimeters).

The absolute units are:

- **in** (inch)
- **cm** (centimeter)
- **mm** (millimeter)
- **pt** (point): 1 inch has 72 points. 1pt is 1/72 in ≈ 0.014in ≈ 0.35mm.
- **pc** (pica): 1 pica is 12 points. 1 inch has 6 picas. 1pc ≈ 1/6 in ≈ 0.17in ≈ 4.2mm. pc is not commonly used.

The relative units are:

- **px**: number of *pixels*, which is relative to and depends on the viewing devices.

For example, suppose that you are using a 17-inch (diagonal) monitor with a resolution of 1024x768. The screen width is about 12.5in. At that resolution, 16px is about 16x12.5/1024 inch ≈ 0.2in ≈ 0.5cm. Note that pt (point) is absolute (because 72 pt is 1 inch) but px (pixel) is relative, which depends on the display devices.

- **%** (percent): in term of the percentage of a property of a referenced element, generally, *the same property of the parent element*.
  For example, table { width:80% } set the table's width to 80% of the width of the parent (probably a or <body>).
- **em**: the width of the letter 'm' of a referenced font, generally, the current font. For example, margin:2em means that the margins are twice the current (referenced) font-size. However, if em is used to set the font-size property, it needs to find a reference. In this case, it is referenced to the parent's font-size. For example, p { font-size:1.2em; } sets the fontsize of <p> to 1.2 times of the parent (possibly a or <body>).
- **ex** (not commonly-used): the height of letter 'x' of the parent's font. ex is not commonly used.

There shall be no space between the number and the unit, as space is used to separate multiple values.

Take note that % and em measurement are relative to another element (percentage values are always relative, e.g., 50% of something). For example,

```
p {
  width: 80%;        /* 80% of the parent's width */   font-size: 1.2em;  /* 1.2 times of the parent's font */   margin: 1.2em;     /* 1.2 times of the current font's letter 'm' */ padding: 10px;     /* 10 pixels */   border: 0;         /* zero does not need a unit */ }
```

To add to the confusion, some properties, such as line-height, can also accept a bare number, without a unit. This bare number is treated as a *factor* to be multiplied by a reference. For example,

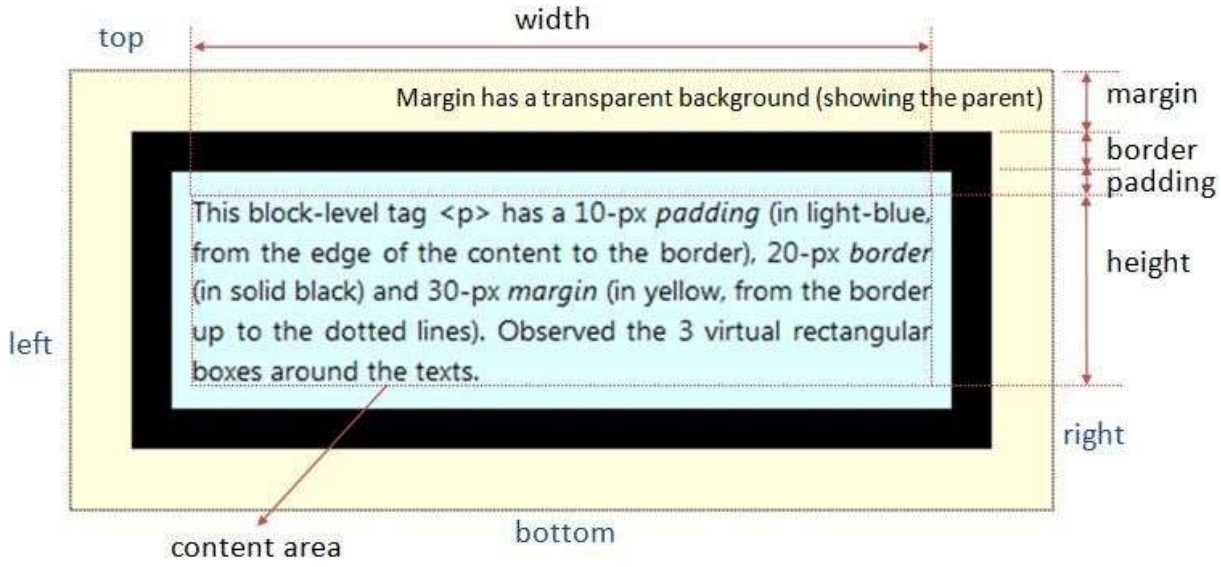
```
line-height: 20px;   /* 20 pixels */ line-height: 150%;   /* 150% of the parent's line-height */ line-height: 1.2em;  /* 1.2 times of the current font's letter 'm' */ line-height: 1.5;    /* 1.5 times of the current font */
```

Take note that in HTML tag attributes, such as width="400", the bare number is measured in pixels.

1.14 Box Model - Margin, Border, Padding and Content Area

Recall that HTML defines two kinds of elements: block element and inline element.

A block element (such as <p>, , <h1> to <h6>) is rectangular in shape and exhibits the so-called *box model*, with *four virtual rectangles* wrap around it "*content area*", representing the *content area*, *padding*, *border*, *margin*, as illustrated below.



1. The *content area* contains the texts, image, or child elements.
2. The *padding* is the space between the content area and the border. It clears an area outside the content area. It has the *same background* as the content area.
3. The border goes between padding and margin. You can set a color and a style (such as solid, dash, dotted) to the border.
4. The margin is the space outside the border (to another element). It clears an area outside the border.
   The margin does not have a background, and is totally transparent.

As illustrated in the box model diagram, margin pushes its border (and content) away with a transparent background showing the parent (having the effect of pushing itself away from the parent); while padding pushes its content inwards with the same background. Margin and padding serve the same purpose if there is no border and background applied.

Take note that the width and height that you set for the element specify its content area, exclude the margin, border and padding. To get the *actual* size of the element, you need to add the margin, border and padding to the width/height. For example, suppose that:

```
#elm {   width:
300px;
margin: 10px;
  border: 5px solid black;   padding:
20px;
}
```

The *actual width* of the element is 300+(10+5+20)x2 = 370px.

Each of the rectangular bounds has four sides, and can be individually referred to as *xxx*-top, *xxx*right, *xxx*-bottom, and *xxx*-left in a clockwise manner, where xxx could be margin, border or padding.
The four sides can be controlled individually or as a group.

CSS Dimension Properties

These properties allow you to set up the dimension, such as the width and height of an element.

- width: auto|*n*|*n*% heigth:
  auto|*n*|*n*%
  The width and height are specified in units such as px (pixels), or *percent* (relative to the parent element).
- max-width: none|*n*|*n*% max-
  height: none|*n*|*n*% min-width:
  none|*n*|*n*% min-height:
  none|*n*|*n*%
  Set the minimum and maximum width and height.

As mentioned earlier, CSS length measurement requires a proper unit, e.g., width:400px or width:80%. Take note that width:400 is meaningless in CSS (this is a very common error!) However, in HTML, width="400" means 400 pixels.
The width and height properties are NOT inherited by its descendants. The default value is "auto", which lets the browser to compute a suitable value. We shall discuss "width:auto" value later.

CSS Margin, Border and Padding Properties

The margin, border and padding related properties are:

- margin-top: auto|*n*|*n*% margin-right: auto|*n*|*n*% margin-bottom:auto|*n*|*n*% margin-left: auto|*n*|*n*%

  Set the four margins individually. The "*n*" shall be expressed in a proper unit (e.g. 10px and 1.2em). You could use a negative value to overlap two elements (e.g., margin-left:-100px). The value of "auto" lets the browser to compute an appropriate number. "*n*%" is relative to the same property (i.e. margin-*xxx*) of the parent.

- margin: *margin-top margin-right margin-bottom margin-lef*t margin: *margin-top-bottom margin-right-left*
  margin: *all-4-margins*

  These are one-line shorthand notations to set all the four margins. If four values are given, they are applied to top, right, bottom, left (in the clockwise manner). If two values are given,

they are applied to top-and-bottom, left-and-right. If one value is given, it is applied to all the four borders.

- padding-top: *n*|*n*% padding-right: *n*|*n*% padding-bottom: *n*|*n*% padding-left: *n*|*n*%
  Set the four paddings individually, similar to margin-*xxx*.

- padding: *padding-top padding-right padding-bottom padding-left* padding: *padding-top-bottom padding-left-right* padding: *all-4-padding*
  A one-line shorthand notation to set all the four paddings, similar to margin.

- border-width: thin|medium|thick|*n*

  Set the width of the four borders. "*n*" can be used to set an absolute thickness. border-width is a shorthand notation, you can use border-width-top, border-width-right, border-width-bottom, and border-width-right to set the four borders individually.

- border-style: none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset

  Set the style of the 4 borders. Similarly, you can use border-style-top, border-styleright, border-style-bottom, and border-style-right to set the four borders individually.

- border-color: *#rrggbb*|rgb(*r*,*g*,*b*)|rgba(*r*,*g*,*b*,*a*)|*color-name*

  Set the color of the 4 borders. Similarly, you can use border-color-top, border-colorright, border-color-bottom, border-color-right to set the four borders individually.

- border: *border-width border-style border-color*

  Shorthand notation to set all the properties of the borders, in the order shown. You can also use the border-top, border-right, border-bottom, and border-left to set the four borders individually.

Margin, border, padding, width are NOT inherited by its descendants.

width:auto

For most of the block elements (e.g., , <p>), the default of width:auto sets the width to the width of the parent minus its own margin, border and padding. Images <img> have an auto width equals to its actual width. Float elements have auto width of 0.

Filling the Width of the Containing Element

Browser would automatically adjust the margin-right to fill the container's width if the sum of its width, left and right margin/border/padding does not add up to the full width of the containing

element. Take note that browser will not adjust the width,padding-right, border-right and the left margin/border/padding.

Center a Block Element

To center a block element, you set the margin-left and margin-right to auto (browser divides the remaining width to left and right margins equally).
For example, all the selected are centered:

```
div#header {
  margin: 10px auto;  /* 20px for top and bottom, auto for left and right */
}
div#footer {
  margin: 10px auto 5px auto /* top right, bottom, left */
}
div#content {
  margin-top: 10px;
  margin-right: auto;
  margin-bottom: 5px;
  margin-left: auto;
}
```

Example

The above diagram was produced using these codes.

 This block-level tag <p> has a 10-px
padding (in light-blue, from the edge of the content to the border), 20-px
border
(in solid black) and 30-px margin (in yellow, from the border up to
the dotted lines). Observed the 3 virtual rectangular boxes around the texts.

1.15 Font Properties

The frequently-used font properties are:

- font-family: *font-name*|*generic-family-name*
  A *prioritized* list of fonts to be used. The browser will try to use the first font if the it is available, and goes down the list. The *generic font family* names include: serif (with small tails), sans-serif (without small tails), monospace, cursive, fantasy. Use monospace for program listing. Use sans-serif for computer display. serif are mainly used in prints (such as "Times" for newspapers and books).
- font-size: *n*|*n*%|xx-small|x-small|small|medium|large|x-large|xxlarge|smaller|larger

- font-weight: normal|bold|bolder|lighter|100|200|...|800|900
  You can use a number between 100 to 900, in multiple of 100. The value of 400 is the normal weight; while 700 is bold.
- font-style: normal|italic|oblique
  The italic uses italic font installed (some font families include the italic version); while the oblique is done by tilting the normal font.
- font-variant: normal|small-caps
  The samll-caps is smaller than the uppercase.
- font: *font-style font-variant font-weight font-size/line-height font-family*
  Set all the font properties using a one-line shorthand notation. The properties must follow the order shown above. However, the leading and trailing items can be omitted. For example,

```
p {
  font-size: 14px;   font-weight:
bold;   line-height: 140%;
  font-family: Arial, sans-serif;
}
```

  is the same as:

```
p { font: bold 14px/140% Arial, sans-serif; }
```

Font properties are inherited by its descendants.

1.16 Text Properties

The frequently used text properties are:

- text-align: left|right|center|justify
- line-height: normal|*n*|*n*%|*factor*
  Set the height of the line. The *factor* gives the factor to be multiplied by the current font-size. E.g,, factor of 1.5 means 1.5 times of the current font.
- text-decoration: none|underline|overline|line-through|blink
  Graphic designer dislikes "underline" and considers it as a legacy of typewriter. "blink" is even worse!
- text-transform: none|uppercase|lowercase|capitalize
  The capitalize transforms the first letter to uppercase.
- text-indent: *n*|*n*%
  Indent the first-line of the paragraph. To indent all the lines of a paragraph (i.e., the whole block), use padding or margin.

- letter-spacing: normal|*n* word-spacing: normal|*n*
  Additional spacing to be applied to letters or words.
- white-space: normal|pre|nowrap
  Specify how white spaces inside the element is to be handled. For "pre" (pre-formatted), preserve the white-spaces.

1.17 Background Properties

The background related properties are:

- background-color: *#rrggbb*|rgb(*r*,*g*,*b*)|rgba(*r*,*g*,*b*,*a*)|*color-name*|transparent
  Set the background color of an element. The default is transparent.
- background-image: url(*imageURL*)|none
  Use an image as the background.
- background-repeat: repeat|repeat-x|repeat-y|no-repeat
  Define how the background image shall be repeated in x and y direction or both.
- background-attachment: scroll|fixed
  Define whether background image shall scroll with the page or fixed.
- background-position: *x y*|*x*% *y*%|top left|top center|top right|center left|center center|center right|bottom left|bottom center|bottom right
  Set the initial position of the background image. Note that there are two values, specifying the x and y position respectively.

The background properties has a one-line shorthand notation, with the order shown as below:

- background: *background-color background-image background-repeat backgroundattachment background-position*

In all the above, the term *background* refers to the background of the elements selected (not necessary the entire window). In other words, you can set an image as the background of an element.

1.18 List Properties

The properties are:

- list-style-type: none|disc|circle|square| list-style-type: lower-alpha|upper-alpha|decimal|decimal-leading-zero|lowerroman|upper-roman|lower-greek|lower-latin|upper-latin
  Set the style of the list item marker for <ol>, <ul> and <li>..
- list-style-position: inside|outside
  Define whether the list item marker shall be inside or outside the item element.

- list-style-image: none|url(*imageURL*)
  Use an image as the list item marker.
- list-style: *list-style-type list-style-position list-style-image*
  Shorthand notation to specify all the properties of the list.

1.19 Table Properties

The properties are:

- border-collapse: collapse|separate
  Collapse or separate the adjacent cells shared border into one.
- border-spacing: *n*
  For separate border, specify the distance between border (i.e., the deprecated cellspacing attribute)
- caption-side: top|bottom|left|right
  Specify which side to show the caption.
- empty-cells: show|hide
- table-layout: auto|fixed

Exercise:

Create a Resume/CV design using External CSS and HTML.
Check this sample from youtube: https://www.youtube.com/watch?v=zAVhHHS_IH4