ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY

Kanyakumari Main Road, near Anjugramam, Palkulam, Anjugramam, Tamil Nadu 629401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATION COURSE

ON

HTML & CSS

COURSE MATERIAL

HTML & CSS

1. Introduction to HTML/CSS

To create an One Man Operated website, We need to

1. Understand HTML, CSS and JavaScript thoroughly.
2. Pick an authoring tool: Use Dreamweaver if you can afford. Otherwise, find a free source-code editor (such as VS Code, Sublime Text, Sublime Text, NotePad++). For programmers, Eclipse/NetBeans/VS Code are good choice for HTML/CSS/JavaScript as they perform syntax checking and provide auto-code-complete.
3. Design and organize your page. Decide on the *look and feel* of your website. How many columns? What are the major sections (e.g., header, navigation menu, main content, sidebar, table of content, footer)? Do you need a navigation menu or panel? What is your theme (colors, fonts)? And so on.
4. Take a close look at your favorite websites!!! CSS is humongous and complex! You can't invent this wheel! Use F12 Debugger to inspect HTML/CSS of your favorite websites. Use a CSS framework, such as BootStrap, to *jump-start* your design.
5. Start with an initial CSS design. Website design begins with CSS, NOT HTML?!. Work on your CSS:
    a. Partition your web page into logical section via (or HTML5' <header>, <footer>, <section>, <nav>), such as header, content, footer. Assign an id to that is unique (e.g., "header", "footer". Assign a common classname to sections (non-unique) that share the same style (e.g., "entry", "side-note"). Write the CSS *id-selectors* and *class-selectors* (e.g., #header *tag-name*,... #footer *tag-name*,... #menu *tag-name*,...) for common tags (such as h1, h2, h3, p, a:link, a:visited, a:hover, a:active), in each of the 's. Basically, what I am saying is to design each of the sections by itself - a "divide and conquer" strategy.
    b. Create sub-classes for common styles, such as layout out tables and images and special effects (e.g., ".highlight", ".underline", ".center"). They could be used in and .

c. There are many good and free CSS templates (or web templates) available online (just google "CSS Templates" or "Web templates"). Pick one that meets your taste to model after. You can also look at the CSS of any website that you find interesting. Be aware of the Intellectual Property Right, do not use any images or graphics unless they are in the public domain. It is extremely easy to create one yourself with an imaging tool, such as PhotoShop, Element, Illustrator or even Paint.
   Use a CSS framework, such as BootStrap; and pick your favorite design from the samples.

6. Write your HTML pages. You may need to modify the CSS as you go along. The most challenging thing for an OMO web author is that he has to be concerned about both the contents and appearances at the same time, and can lose focus at times!
7. Repeat the previous steps until you are happy with your page's look and feel, layout, and most importantly, the contents - try not to create *yet another insignificant* website.

I wrote the above many years ago while creating these web pages (You can check out my CSS). Today, I would recommend that you learn the basics of HTML/CSS, but quickly jump into a framework such as BootStrap to produce professional-looking web pages. You can't write better than these people!

2. Introduction to HTML

What is HTML (HyperText Markup Language)?

1. HTML is the *language* for publishing web pages on the WWW (World-Wide Web, or World-Wide Wait?).
2. HTML is a *Document Description Language* (aka *Document Markup Language*). HTML is the standard markup language for documents designed to be displayed in a web browser. HTML is NOT a programming language like C/C++/C#/Java, which is used to implement programming algorithm.
3. An HTML document is a text document, and it is human-readable.
4. Today, HTML works together with CSS (Cascading Style Sheets) (for layout) and JavaScript (for programming).

HTML Versions

- HTML Draft (October 1991): Tim Bernes-Lee (of CERN) proposed the early HTML (with 18 tags) for sharing of document in a hypertext system.
- HTML 2.0 (November 24, 1995): Published as IETF RFC 1866.
- HTML 3.2 (January 14, 1997): Published as W3C HTML 3.2 Recommendation.
- HTML 4.0 (December 1997): Published as W3C HTML 4.0 Recommendation, with strict, transitional and frameset. In December 24, 1999, HTML 4.01 was published as the *final* HTML specification by W3C. In May 2000, HTML 4.01 Strict as published as ISO/IEC International Standard 15445:2000.
- XHTML 1.0 (January 2000): W3C considered HTML 4.01 as the final HTML, and moved on to develop XHTML 1.0 with stricter rules and syntaxes, followed by XHTML 2.0. XHTML 2.0, although theoretically elegant, is impractical as it is not backward compatible with HTML4/XHTML1.0. A rebel group called WHATWG (Web Hypertext Application Technology Working Group) continued to work on extending HTML with more features in a backward-compatible manner. In 2004, WHATWG released HTML5. By 2007, HTML5 has captured the attention of the developers. W3C decided to abandon the XHTML 2.0 and embraced the HTML5.
- HTML 5 (October 28, 2014): HTML 5 was published as W3C Recommendation, followed by HTML 5.1 on November 1, 2016, and HTML 5.2 in December 14, 2017.
- On 28 May 2019, the W3C announced that WHATWG would be the sole publisher of the HTML and DOM standards.

Today, the prevailing specifications are HTML5 (@ https://html.spec.whatwg.org/multipage/). Nonetheless, the most interesting thing about standards is that nobody really follows them strictly. Every browser (Chrome, Firefox, Opera, Safari and Internet Explorer) has its own variations and support the standards to various extents (so as to out-do others).

Markup Tags

HTML uses *markup tags*, such as <p> (for Paragraph), <h1> to <h6> (for Heading Level 1 to 6), <img> (for Image), <a> (for Anchor or Hyperlink), to markup a document. HTML markup tags perform these functions:

1. Layout the documents, e.g., <p> (layout as a paragraph), <h1> to <h6> (layout as heading level 1 to 6),
 (perform a line break), <hr> (draw a horizontal rule), <table> (tabulating data), <ol> (layout an ordered list).
2. Provide link (called *hyperlink*) to another HTML document, via the <a> (Anchor tag). These hyperlinks, a distinct feature in HTML, greatly help the users in navigating the web and enrich the users' experience. Hyperlinks make the HTML popular.
3. Embed images, audios, videos, programs (in JavaScript, VBScript, Applet, Flash, or MS ActiveX control), and objects within an HTML document. HTML is *multimedia*! The hypertext document may contain texts, images, audios, videos, and even programs.

Separating Content and Presentation

The purpose of a markup language is to relieve the content provider from worrying about the actual appearance of the document. The author merely indicates (via markup tags) the *semantic meaning* of the words and sentences (such as paragraph, heading, emphasis, and strong), and leave it to the browser to interpret the markups and render the document for display on the screen. In other words, it allows the *separation of content and presentation*. The content provider focuses on the document contents, while the graphic designer concentrates on the view and presentation.

Nowadays, HTML should be used solely to markup the contents, while its companion technology known as CSS (Cascading Style Sheet) be used for defining the presentation of the document, so as to *separate the content and presentation*.

These are the common pitfalls in *older* HTML documents and you should avoid:

- Do not specify "appearance" properties, such as foreground and background color, text alignment, font face, font size, border, margin and padding, in the HTML document. Use an external CSS instead to set the appearance and presentation. Presentation-related tags (such as <font>,) and attributes (such as align, bgcolor, link, vlink, alink) have been deprecated in HTML 4, in favor of CSS.
- Do not use nested tables or frame for formatting the document, use and , or HTML5 new tags such as <header>, <footer>, and <section>.

3. HTML By Examples

Let's go thru some HTML examples and the syntaxes. Do not attempt to start designing your website until you understand the CSS.

3.1 Example 1: Basic Layout of an HTML Document

Let's begin with a simple example to illustrate the *basic layout* of an HTML document.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Basic HTML Document Layout</title>
</head>
<body>
  <h1>My First Web Page</h1>
  <hr>
  <p>This is my <strong>first</strong> web page written in HTML.</p>
  <h3>HTML</h3>
  <p>HTML uses <mark>markup tag</mark> to <em>markup</em> a document.</p>
</body>
</html>
```

How it Works?

1. An HTML document begins with a *Document-Type Declaration* <!DOCTYPE html> (Line 1) to identify itself as an HTML document to the browser.
2. The HTML content is contained within a pair of <html>...</html> *container tags*. You can specify the *default language* of your document via *attribute* lang="en" (for English) inside the <html> opening tag.

3. There are two sections in the document: HEAD and BODY, marked by <head>...</head> and <body>...</body> tags, respectively.
4. **In the HEAD section:**
    a. The <meta charset="utf-8"> element (Line 4) specifies the *character encoding scheme* of the document. Today, virtually all (English) HTML documents are encoded using the UTF-8 character encoding scheme, which is compatible with ASCII code for English alphabets and allow you to include other Unicode characters (such as Chinese, Japanese and Korean) efficiently.

When saving your file, you need to choose "UTF-8 encoding" in the "save-as" dialog menu.

b. The <title>...</title> element (Line 5) provides a *descriptive title* to the page. The browser displays the title on the title-bar of the tab/window.

5. **In the BODY section:**
   a. The <h1>...</h1> *container* tags (Line 8) mark the enclosing texts as Level-1 Heading. There are six levels of heading in HTML, from <h1>...</h1> (largest) to <h6>...</h6>. Line 11 uses a <h3>...</h3> (Heading Level-3).
   b. The <hr> *standalone* element (Line 9), which does not enclose text content, draws a horizontal rule (or line).
   c. The ... container tags (Line 10 and 12) mark the enclosing texts as a paragraph. ... is the most frequently-used tag in HTML.
   d. The ... tags (nested under the ... in Line 10) specify "strong emphasis" for its content - rendered in bold by the browser. Similarly, the nested ... tags (Line 12) specify "emphasis" rendered in italic; and <mark>...</mark> (HTML 5) requests for highlight.

View Page Source

You can read the HTML source code by right-clicking on the page and select "View Source" (or "View Page Source", or "Show Page Source"). Try it out.

Note: For macOS's Safari, you may need to enable "Show Page Source" via "Preferences" ⇒ Advanced ⇒ "Show Develop menu in menu bar".

Don't Load the Cached Page (Ctrl-F5)!

Most browsers *cache* web pages (and the associated images, style sheets, JavaScripts) to boost the speed for web surfing. When you modify and reload a page (via the refresh key F5), the browser might retrieve the un-modified cache copy. To force the browser to reload the page (and all its associated resources) from the source, hit Ctrl-F5.

Most Frequently-Used HTML Elements

The most frequently-used HTML elements are:

- block elements: (paragraph),
 (line break), <h1> to <h6> (heading level 1 to 6), <hr> (horizontal rule), (unordered list), (ordered list).

- inline elements: <b> (bold), <i> (italic), <img> (image), <a> (anchor for hyperlink).
- table <table><tr><th><td>.

We shall illustrate the use of these elements through the following examples.

3.2 Example 2: Lists and Hyperlinks

```
<!DOCTYPE html>
<html lang="en">
<!-- Save as "HtmlEg2.html" -->
<head>
  <meta charset="utf-8">
  <title>Lists and Hyperlinks</title>
</head>
<body>
  <h1>Lists and Hyperlinks</h1>
  <p>There are two types of <em>lists</em> in HTML:</p>
  <ol>
    <li>Ordered List.</li>
    <li>Unordered List.</li>
  </ol>

  <p>This is a nested unordered list of links:</p>
  <ul>
    <li>Online Validator:
      <ul>
        <li>W3C Online HTML Validator @ <a
href="https://validator.w3.org/">https://validator.w3.org/</a>.</li>
        <li>W3C Online CSS Validator @ <a href="https://jigsaw.w3.org/css-
validator/">https://jigsaw.w3.org/css-validator/</a>.</li>
      </ul>
    </li>
    <li>Specifications:
      <ul>
```

```
      <li>HTML5 @ <a
href="http://www.w3.org/TR/html5/">http://www.w3.org/TR/html5/</a>.</li>
      <li>CSS3 Selectors @ <a href="http://www.w3.org/TR/css3-
selectors/">http://www.w3.org/TR/css3-selectors/</a>.</li>
    </ul>
   </li>
  </ul>
</body>
</html>
```

How it Works?

1. The <!-- ... --> (in Line 3) is an HTML comment. Comments are ignored by the browsers, but are important to provide explanations to the readers as well as the author.
2. There are two types of lists in HTML: ordered list and unordered list. An ordered list is marked by <ol>... and displayed with numbers; while a unordered list is marked by <ul>... and displayed with bullets. Each of the list items is marked by <li>....
3. You can *nest* a list inside another list, by placing the complete inner list definition inside a list item <li>... of the outer list.
4. Hyperlink is marked by <a> standalone tag. The attribute href="*url*" provides the destination URL of the link.

3.3 Example 3: Tables and Images

```
<!DOCTYPE html>
<html lang="en">
<!-- Save as "HtmlEg3.html" -->
<head>
<meta charset="utf-8">
<title>Table and Images</title>
<style>
table {  /* table */
  border: 1px solid black;
  border-spacing: 5px;
```

```
  border-collapse: separate;
}
th, td {  /* cells */
  border: 1px solid #aaa;
  padding: 5px 10px;
}
</style>
</head>
<body>
 <h1>Table and Images</h1>
 <table>
   <caption>Logo of Languages</caption>
   <tr>
     <th>S/No</th>
     <th>Language</th>
     <th>Logo</th>
   </tr>
   <tr>
     <td>1.</td>
     <td>HTML5</td>
     <td><img src="../images/HTML5_Logo_128.png" alt="HTML Logo" height="64"
width="64"></td>
   </tr>
   <tr>
     <td>2.</td>
     <td>CSS3</td>
     <td><img src="../images/css3.png" alt="CSS Logo"  height="64" width="64"></td>
   </tr>
   <tr>
     <td>3.</td>
     <td>JavaScript</td>
     <td><img src="../images/js3.png" alt="JavaScript Logo" height="64" width="64"></td>
```

```
  </tr>
 </table>
</body>
</html>
```

How it Works?

1. A table, consisting of rows of cells, can be marked via <table>...</table>.
2. A HTML table is row-centric. You shall first mark a row via <tr>...</tr>, and then mark the cells of the row via <th>...</th> (for header cell) or <td>...</td> (for details cell).
3. The <caption>...</caption> element can be nested under <table> to provide a caption for the table.
4. Image is marked via the <img> tag. The mandatory attribute src specifies the path (or url) for the image source file; alt gives the alternative text if the image cannot be displayed. I used relative path in the src, where ".." denotes the parent directory. You need to find some images, store them and figure out your own relative path.
5. The <img>'s optional attributes width and height specify the width and height of the image displayed area. They are used here to resize the images for consistent display.
6. In the HEAD section, I added some so-called *style rules* under the <style>...</style> tags, so as to nicely display the table. You could ignore the styles now, which will be covered later in the CSS section.

3.4 HTML Template

HTML Document Template

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <title>YOUR TITLE HERE!</title>
 <link href="filename.css" rel="stylesheet">
 <script src="filename.js"></script>
</head>
```

```
<body>
  <!-- YOUR CODE HERE! -->
</body>
</html>
```

External CSS and JavaScript are often used in an HTML document. Line 6 includes an external CSS file; and line 7 includes an external JavaScript file.

3.5 HTML Document Validator

You can submit your HTML document to the W3C Online Validator (@ https://validator.w3.org/) for validation.

3.6 Debugging HTML

F12 Debugger (Web Developer Tools)

Recall that you can right-click ⇒ View Source (or View Page Source) to inspect the raw HTML markups.

In most of the browsers, you can push F12 to get into *Web Developer Tools*, which support debugging HTML, CSS, JavaScript, DOM and more.

To debug HTML:

1. Choose the "Inspector" panel to view your HTML codes.
2. To inspect an element, click on the "picker" and point to the HTML element of interest. You can check/modify the "Style", "Layout", "DOM" and "Events" (on the right panel) associated with the selected element.

Try it out on the earlier examples.

4. HTML Basics

HTML Tags

An HTML *opening tag* is enclosed by a pair of angle brackets in the form of <*tag-name*> (e.g., , <title>), which is associated with a matching *closing tag* </tag-name> having a leading forward slash, (e.g., , </title>). The *tag-name* shall be in lowercase.

Tag's Attributes

Attributes, in the form of name="value" pairs can be included in the *opening tag* to *provide additional information* about the element.

Example 1: In <html lang="en">, the attribute lang="en" specifies the natural language for this document.

Example 2: In <meta charset="utf-8">, the attribute charset="utf-8" specifies the character encoding scheme.

Example 3: An <img> (image) tag may contain these attributes:

```
<img src="logo.gif" alt="logo" width="50" height="30" >
```

The attribute "src" specifies the source-URL of the image; "alt" specifies an alternate text, if the image cannot be displayed; "width" and "height" specify the width and height of the image displayed area.

Some of the attributes are *mandatory* (e.g., the "src" and "alt" attributes of the <img> tag); while some are *optional* (e.g., the "width" and "height" attributes of the <img> tag, which are used by browser to reserve space for the image and resizing the image; but browser can figure out the width and height after the image is loaded).

Multiple attributes are separated by space, as follows:

```
<tag-name attName1="attValue1" attName2="attValue2" ...> ... </tag-name>
```

Attributes are written in the form of *name*="*value*" pairs. The *value* should be enclosed in single or double quotes for XHTML/XML compliance (although HTML5 does not enforce it).

HTML Elements

An HTML *element* consists of the opening and closing tags, and the content in between, e.g., A for apple , Caution!.

There are two types of elements:

1. Container Element: A container element has an *opening tag* <*tag-name*> that activates an effect to its content, and a *matching closing tag* </*tag-name*> to discontinue the effect. In other words, container elements apply formatting to their contents. For example:

```
<h1>The h1 tags enclose a heading level 1</h1>
<p>The p tags is used to <em>markup</em> a <strong>paragraph</strong>.</p>
```

2. Void Element (or Standalone Element or Empty Element): A void element does not enclose content but is used to achieve a certain effect, e.g., <hr> is used to draw a horizontal rule;
 to introduce a manual line-break; and <img> for embedding an external image. In **XHTML**, you need to end a standalone element with a trailing '/' in the opening tag. For examples:

```
<br />
<hr />
<img src="logo.gif" />
```

Alternatively, you can also close a standalone element with a matching closing tag. This is cumbersome but consistent in syntax to the container element. For example,

```
<br></br>
<hr></hr>
<img src="logo.gif"></img>
```

HTML5 no longer enforces the above rules. Today, most developers omit the trailing '/' for brevity, e.g.,
, <hr>.

HTML4 has these void elements: <area>, <base>,
, <col>, <hr>, <img>, <input>, <link>, <meta>, <param>.

HTML5 added <command>, <keygen>, <source>.

Nesting Elements and the Document Tree

An HTML element can be nested within another element, e.g., This is my first web page! . It is important to *properly* nest the elements, e.g., the sequence .... is invalid.

A valid HTML document exhibits a *tree* structure (called DOM or *Document Object Model*), with the element <html> as the *root* element of the document tree, with two child elements: <head> and <body>.

[TODO] Draw the DOM tree for one of the examples.

Rendering HTML Pages

Browsers follow these rules when rendering HTML documents:

1. In HTML5, the tag-names are NOT case sensitive. I recommend using lowercase, as it is easier to type and XHTML compliance. (XHTML/XML is case sensitive.)
2. Blanks, tabs, new-lines and carriage-returns are collectively known as *white spaces*. "Extra" white spaces are ignored. That is, only the first white space is recognized and displayed. For example,

```
<p>See   how the   extra white    spaces,
    tabs  and
  line-breaks    are   ignored  by  the
        browser.</p>
```

produces the following single-line output on screen with words separated by a single space:

```
See how the extra white spaces, tabs and line-breaks are ignored by the browser.
```

We need to use the paragraph tag ... to lay out a paragraph; or insert a manual line-break tag
 to break into a new line. In other words, you control the new-lines via the mark-up tags. The ... leaves additional space after the paragraph; while
 does not.

To get multiple whitespace, use a special code sequence   (which stands for non-breaking space). For example,

```
<p>This is a
paragraph.</p>
```

```
<p>Another paragraph<br>with a line-break in between and more spaces     before this.</p>
```

This is a paragraph.

Another paragraph
with a line-break in between and more spaces before this.

4.2 The Basic Document Skeleton

The <html> **Element and its Child Elements** <head> **and** <body>

An HTML document has the following skeleton:

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>...</body>
</html>
```

The <html>...</html> container element defines the extent of an HTML document.

There are two sections: HEAD and BODY:

1. <head>...</head> container element defines the HEAD section, which contains *descriptions* and *meta-information* of the HTML document. Browsers interpret these descriptions to properly display the body content.
2. <body>...</body> container element defines the BODY section, which encloses the *content* that users actually see in the browser's window.

The HEAD **Section and the** <title>**,** <meta>**,** <link>**,** <script> **Elements**

The HEAD section may contain these elements: <title>, <meta>, <link>, <script>, <base>.

The <title>...</title> container element encloses the *title* for the page. You should use a meaningful title because:

- The title shows up at the title-bar of the browser window.
- The title shows up in bookmarks and history lists (the URL is stored if there is no title).
- Titles are used by search engines to index your page.

The <title> element is mandatory in HTML5.

The <meta> element contains meta-information, for use by browser to properly render the document. For example, <meta charset="utf-8"> specifies the character encoding scheme for the document.

You can use a <link> element to add an external CSS Style Sheet (and <style>...</style> element for internal styles):

```
<link href="filename.css" rel="stylesheet">
```

Note: HTML4/XHTML1 require an additional attribute type="text/css" in the <link> tag.

You can use a <script>...</style> element to define programming scripts. For example, to add an external JavaScript file:

```
<script src="filename.js"></script>
```

Note: HTML4/XHTML1 require an additional attribute language="JavaScript" in the <script> tag. Also take note that the closing </script> tag is needed, even though there is no content within <script> and </script>. We usually place the JavaScript after CSS, as the JavaScript may use the CSS.

The <body> **Element**

The <body>...</body> element defines the BODY section of an HTML document, which encloses the content to be displayed on the browser's window.

The <body> tag has the following optional presentation attributes. All of these presentation attributes are concerned about the appearance instead of the content, and have been deprecated in HTML 4 in favor of style sheet. However, many older textbooks present them in Chapter 1. Hence, I shall list them here for completeness. BUT do not use these attributes. I shall describe how to control the appearance of <body> using CSS very soon.

- text="*color*": color of body text.
- bgcolor="*color*": background color.
- background="*url*": URL of an image to be used as the background.
- link="*color*": color of un-visited links.
- vlink="*color*": color of visited links.
- alink="*color*": color of active (clicked) links.

For example:

```
<html>
<body text="blue" bgcolor="lightblue" link="green" vlink="red" alink="yellow">
```

```
  <p>Hello</p>
  <a href="http://www.google.com">Google</a>
</body>
</html>
```

The foreground color (of the texts) is "blue", on background color of "lightblue". You can set different colors for the three types of links via attributes "link" (for un-visited links), "vlink" (for visited links), and "alink" (for active link - the "alink" color shows up when you click on the link).

4.3 HTML Comment <!-- ... -->

HTML comments are enclosed between <!-- and -->. Comments are ignored by the browser. You need to look into the HTML codes (via "View Source" or "View Page Source") to read them. Comments are extremely important in *programming* to explain and document a section of programming codes and logic. HTML documents are textual and self-explanatory, comments are less important (but still nice to have) to describe the various part of the documents.

Comments are also useful in temporarily disable a certain part of the HTML codes during development.

You can comment out an entire elements, e.g.,

```
<!--
<h1>hello, world</h1>
-->
```

But you cannot place comment inside a tag, e.g.,

```
<h1 <!-- invalid comment -->>hello, world</h1>
```

4.4 Block Elements vs. Inline Elements

Elements can be classified as:

1. Block Elements: A block element (such as <p>, <h1> to <h6> and) starts on a new line, takes the full width, and ends with a new line. It is *rectangular* in shape with a *line-break* before and after the element.
2. Inline Elements (or Character Elements): An inline element (such as , , <code> and) takes up as much space as it needs. It does not force a line-break before and after the element, although it can span a few lines.

In brief, a block element is always *rectangular* in shape, while an inline element spans *a continuous run of characters*.

4.5 Block Elements

Paragraph ...

Function: When the browser reads a tag, it breaks to a new line, and skips some white spaces. For example,

```
<p>This is a paragraph of texts.</p>
```

Older HTML documents often omit the closing , which is a bad practice, not recommended, and disallowed in XML/XHTML.

To create an empty paragraph, you need to use   which encloses a non-breaking space. This is because browsers typically ignore .

Line Break
 (Standalone)

Function: Instruct the browser to break to a new line, without skipping white spaces as in . Note that the line breaks in the HTML codes are treated as white spaces and do not translate to new lines in the display. Hence, you have to insert the
 manually. For example,

```
<p>This
paragraph<br>with a
line-break
in between.</p>
```

This paragraph
with a line-break in between.

Heading Level 1 to 6 <h1>...</h1> **to** <h6>...</h6>

Function: Establish six levels of document headings. <h1> is the highest (with the largest font size) and <h6> is the lower. Headings are usually displayed in bold, have extra white spaces above and below. For example,

```
<h1>This is Heading Level 1</h1>
<h2>This is Heading Level 2</h2>
<h3>This is Heading Level 3</h3>
<h4>This is Heading Level 4</h4>
<h5>This is Heading Level 5</h5>
<h6>This is Heading Level 6</h6>
```

```
<p>This is a paragraph</p>
```

Horizontal Rule <hr> **(Standalone)**

Function: Draw a horizontal line (or rule). By default, the rule is full width (100%) across the screen, 1 point in size, and has a shading effect for a 3D appearance. For example,

```
<h1>Heading</h1>
<hr>
<p>Discussion begins here...</p>
```

Pre-Formatted Text <pre>...</pre>

Function: Texts enclosed between <pre>...</pre> container tags are treated as *pre-formatted*, i.e., white space, tabs, new-line will be preserved and not ignored. The text is usually displayed in a fixed-width (or monospace) font. <pre>...</pre> is mainly used to display program codes. For example, my favorite Java's "Hello-world":

```
<pre>public class Hello {
  public static void main(String[] args) {
    System.out.println("Hello");
  }
}</pre>
```

```
public class Hello {
  public static void main(String[] args) {
    System.out.println("Hello");
  }
}
```

Without the <pre> tag, the entire program will be shown in one single line.

Quote <blockquote>...</blockquote>

Function: Mark out *a block of quote*. Browsers typically indent the entire block to the right. For example,

```
<blockquote>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod 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
```

```
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
 Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.</blockquote>
```

Division ... **and Span** ...

The block elements, together with its inline counterpart , are extensively used in the modern web pages to mark out a rectangular block or inline span of text. The and , by itself, does not do anything. Unlike elements such as <h1>, <p> and , which possess certain visual properties, and do not have any inherent visual properties. They are simply markers and to be used together with CSS for applying custom formatting style. We shall describe them later in the CSS section.

Best Practices:

- Notes: With the introduction of many new semantic block elements in HTML5, such as <header>, <footer>, <main>, <section>, it is strongly encouraged "to view the element as an element of last resort, for when no other element is suitable. The use of more appropriate elements instead of the element leads to better accessibility for readers and easier maintainability for authors".
- Avoid deep nested container layouts using . "should be not nested deeper than 6 layers".
- Use as little HTML elements as possible (to increase the content to tag ratio and reduce page load).

4.6 HTML5's New Semantic Block Elements

Before HTML5, we rely on the all-purpose generic container elements to structure a document into various sections and apply the formatting style, For example,

```
<div class="header">
  ......
</div>


<div class="content">
  ......
</div>
```

```
<div class="footer">
  ......
</div>
```

This is less than desirable, as elements do not provide semantic information about the sections.

HTML5 added many semantic block elements, which extends to structure a document. They are: <header>, <footer>, <nav>, <section>, <article>, <summary>, <details>, <aside>, <figure>, <figcaption>, and <main>. You are encouraged to replace some of the 's with these more descriptive semantic elements. Nonetheless, it is important to note that NOT all browsers (notably older IE versions) support these new elements.

Header <header>...</header>**, Footer** <footer>...</footer> **and Section** <section>...</section>

The <header>...</header> and <footer>...</footer> elements can be used to markup the header and footer of a web page, in place of the less semantic pre-HTML5 <div id|class="header"|"footer">.

The <section>...</section> element can be used to markup each content section in a document (such as each chapter of the book). (HTML5 does not define a <content> element!) For example,

```
<header>
  ......
</header>

<section>
  ......
</section>

<section>
  ......
</section>

<footer>
  ......
```

```
</footer>
```

Article <article>...</article>

The <article>...</article> element is used to markup an *independent* and *self-contained article* such as a news story, which could have its own header, footer and content sections.

Figure <figure>...</figure> **and Figure Caption** <figcaption>...</figcaption>

You can markup a figure via <figure>...</figure> and <figcaption>...</figcaption>, e.g.,

```
<figure>
  <img src="...." alt="...">
  <figcaption>.....</figcaption>
</figure>
```

You can remove the alt attribute from the <img> tag, as <figcaption> typically provides better description.

In a web page, figures are typically *float* alongside the text. For example, you can apply the following style rules to <figure> and <figcaption> to float the figure to the left:

```
figure {
  float: left;
  margin-left: 0px;
  margin-top: 0px;
  margin-right: 20px;
  margin-bottom: 0px;
}

figcaption {
  font-size: small;
  font-style: italic;
  margin-bottom: 5px;
}
```

Sidebar <aside>...</aside>

The <aside> element can be used to introduce related contents, typically formatted in a floating sidebar alongside the main texts.

Navigation Menu <nav>...</nav>

The <nav>...</nav> element wraps a set of links into a navigation menu. For example,

```
<nav>
  <h1>....</h1>
  <ul>
    <li><a href="...">.....</a></li>
    <li><a href="...">.....</a></li>
    .....
  </ul>
</nav>
```

You can place the <nav> under an <aside> if the navigation menu is to be shown in a sidebar (or side panel).

Which Element to Use?

The HTML5 Doctor provides a nice flowchart for you to decide which HTML5 element to use @ http://html5doctor.com/downloads/h5d-sectioning-flowchart.pdf.

Summary <summary>...</summary> **and Details** <details>...</details>

The <summary> and <details> elements are meant for showing the summary and details, typically in a *collapsible box*. You can use the following JavaScript to show/hide the box:

```
var box = document.getElementByID("...");
// Hide the box
box.style.display = "none";
// Show the box
box.style.display = "block";
```

However, the browser supports for these two tags are poor, and it is best to avoid them.

Main Content <main>...</main>

The <main> element (introduced in HTML5.1) marks the *main content* of a web page, excluding the header, footer, and navigation menu. There shall NOT be more than one <main> element in a document. For example,

```
<header>.....</header>

<main>
  <article>.....</article>
    <section>.....</section>
    <section>.....</section>
  <article>.....</article>
</main>

<footer>.....</footer>
```

The <main> element shall NOT be a descendant of an <article>, <aside>, <footer>, <header>, or <nav> tags.

Providing Backward-Compatibility

Chrome, Firefox, Safari and Opera have no problems with these HTML5 tags, so does IE (Internet Explorer) 10. However, IE 9 and IE 8 may have problems rendering these tags.

To provide compatibility to older browsers, you could:

1. Add a CSS rule to render these tags as *block* element (does not work for IE 8):

```
article, aside, figure, figcaption, footer, header, main, nav, section, summary {
  display: block;
}
```

4. Use the HTML5 Shiv (@ https://github.com/afarkas/html5shiv), which contains JavaScript to create these elements.
5. Use Modernizr (@ http://modernizr.com/) - a JavaScript library that detects HTML5/CSS3 features.

4.7 Inline Elements - Logical Style vs. Physical Style

Logical-style formatting tags specify the *semantic meaning* (e.g., strong, emphasis, code); whereas physical-style formatting tags define the *physical* or *typographical appearance* (e.g., bold, italic, teletype). Logical styles should be used instead of physical styles. This is because physical styles

deal with the appearance, which should be defined in style sheet, so as to separate the content and presentation.

Logical-Style Formatting Tags

The logical style character-level (inline) tags are:

Logical-Style Tag	Meaning
<mark>...</mark> (HTML 5)	Highlight
...	strong emphasis (bold)
...	emphasis (italic)
<code>...</code>	program code (fixed-width monospace font)
<q>...</q>	quotation (enclosed in curly double quotes)
<ins>...</ins>	inserted
~~...~~	deleted
<def>...</def>	definition (bold or bold-italic)
<cite>...</cite>	citation (italic)
<kbd>...</kbd>	Keyboard (fixed-width monospace font)
<samp>...</samp>	sample text (fixed-width monospace font)
<abbr>...</abbr>	abbreviation (dotted underline, with title as tool tip)
<acronym>...</acronym>	acronym (dotted underline, with title as tool tip). Recommend usin
<address>...</address>	address

<var>...</var>	variable (fixed-width or italic)

The commonly-used tags are: (displayed in bold), (displayed in italics), and <code> (use monospace font for programming codes).

Example:

```
<p>Lorem <q>curly quoted</q>, consectetur adipisicing elit,
sed do <cite>citation</cite> incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis <samp>sample</samp> exercitation ullamco laboris nisi
ut <code>code</code> ex ea <kbd>keyboard</kbd> consequat. Duis aute irure dolor in
reprehenderit in velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur <ins>insert</ins> occaecat <del>delete</del> non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
```

The contents are shown with a dotted underline. By including an attribute title="*fulltext*" to the <abbr> and <acronym> opening tag, the full text will be shown as *tool tip*, when you point your mouse pointer to the element. HTML5 supports <abbr>, but does not support <acronym>.

For example,

```
<p>Lorem <abbr title="abbreviation">abbr</abbr> dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor <acronym title="Hypertext Markup Language">HTML</acronym> ut l
abore
et dolore magna aliqua.</p>
```

Lorem abbr dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor HTML ut labore e t dolore magna aliqua.

The title="*tooltip-text*" attribute is actually applicable to almost all of the HTML tags (e.g., <p>, <h1> to <h6>, <img>).

Physical-Style Formatting Tags

All the physical-style character-level tags are deprecated in HTML 4, as they deals with presentation and should be done via CSS. I list them here for completeness, in case you need to read some old HTML codes.

However, theHTML5 restores the , , <small>, <sup>, <sub>, <u> tags. The <big>, <tt> remains unsupported.

Physical-Style Tag	Meaning
...	bold
...	italic
<u>...</u>	underline (deprecated in HTML 4, but reinstated in HTML 5)
<big>...</big>	large font
...	small font
^{...}	superscript
_{...}	subscript
<tt>...</tt>	teletype (typewriter, in monospace font)

Span ...

Similar to its block-level counterpart , elements are extensively used in the modern web pages to mark out a run of texts, primarily for applying style.

4.8 HTML5's New Semantic Inline Elements

HTML5 puts back the , and <small> elements (that were deprecated in HTML4/XHTML1), but not the <u> element. It also added more semantic character-level elements for text: <mark>, <time>, <wbr>. These elements are hardly used due to poor browser support, but presented here for completeness.

DateTime <time>

For marking up date, time, or datetime.

Word Break <wbr>

[TODO]

Marked Text <mark>

Highlighting certain texts, such as keywords. By default, it is shown with bright yellow background, similar to highlighter's marking.

4.9 Entity References for Reserved & Special Characters

HTML uses characters such as <, >, ", & as markup tags' delimiters. Hence, these characters are reversed and cannot be used in the text directly. An escape sequence (called *entity reference*), in the form of &*xxx*; (begins with & and ends with ; and contain the code *xxx*) is used for these reserved characters and other special characters. The commonly used entity references are as follows (there are many many more, refer to the HTML reference - I like the arrows, Greek symbols, and the mathematical notations). Entity reference is case sensitive.

4.10 Lists

List-related tags are meant for marking up *a list of items*. HTML supports three types of lists: *ordered list*, *unordered list* and *definition list*.

Unordered List ... **and List Item** ...

Function: An unordered list is shown with a *bullet* in front of each item. The <ul>... contains an unordered list. Each of items in the list is enclosed in <li>..., as follow:

```
<ul>
  <li>list-item-1</li>
  <li>list-item-2</li>
  ......
</ul>
```

Example:

```
<p>An OMO web designer must master:</p>
<ul>
   <li>Hypertext Markup Language (HTML)</li>
   <li>Cascading Style Sheet (CSS)</li>
   <li>HyperText Transfer Protocol (HTTP)</li>
   <li>Apache HTTP Server</li>
</ul>
```

Output of the example:

An OMO web designer must master:

- Hypertext Markup Language (HTML)

- Cascading Style Sheet (CSS)
- HyperText Transfer Protocol (HTTP)
- Apache HTTP Server

You can use attribute type in <ul> tag to choose the style of the bullets:

- type="disc": a black dot (default).
- type="circle": an empty circle.
- type="square": a filled square.

Ordered List ... **and List Item** ...

Items in an ordered list are *numbered* automatically by the browser. The container tag <ol>... contains an ordered list. Each item of the list is contained inside a <li>... container tag. The syntax is similar to the unordered list.

You can use attribute start="*number*" in the <ol> tag to specify the starting number (which default to 1).

You can use the type attribute of the <ol> tag to choose the numbering style:

- type="1": numbers 1, 2, 3, ... (default)
- type="a": lowercase letters a, b, c, ...
- type="A": uppercase letters A, B, C, ...
- type="i": lowercase Roman numerals i, ii, iii, iv, ...
- type="I": uppercase Roman numerals I, II, III, IV, ...

Definition List (or Description List) <dl>...</dl>**, Definition Term** <dt>...</dt> **and Definition Detail** <dd>...</dd>

Function: <dl>...</dl> tag contains a *Definition List*. Each of <dt>...</dt> and <dd>...</dd> pair contains a *Definition Term* and the *Definition Detail*. HTML 5 call it *Description List*.

```
<dl>
  <dt>term-1</dt>
  <dd>definition-for-term-1</dd>
  <dt>term-2</dt>
  <dd>definition-for-term-2</dd>
  .....
</dl>
```

Example:

```
<p>These are some of the commonly-encountered web protocols:</p>
<dl>
  <dt>HTTP</dt><dd>HyperText Transfer Protocol</dd>
  <dt>FTP</dt><dd>File Transfer Protocol</dd>
  <dt>SMTP</dt><dd>Simple Mail Transfer Protocol</dd>
  <dt>POP</dt><dd>Post Office Protocol</dd>
</dl>
```

These are some of the commonly-encountered web protocols:

HTTP

: HyperText Transfer Protocol

FTP

: File Transfer Protocol

SMTP

: Simple Mail Transfer Protocol

POP

: Post Office Protocol

The "unordered list" and "ordered list" are used in most of the HTML documents. But I don't find many web pages using the "definition list".

Nested Lists

You can place a list inside another list (called nested lists), by writing a complete list definition under an <li> item of the outer list. You can nest any types of lists (ordered list, unordered list).

Example 1:

```
<p>The topics covered are:</p>
<ul>
  <li>HyperText Markup Language (HTML)
    <ul>
      <li>Based on SGML</li>
```

```
      <li>Used to create web pages</li>
      <li>Maintained by W3C</li>
    </ul>
  </li>
  <li>Cascading Style Sheet (CSS)
    <ul>
      <li>Used to define presentation style for web pages</li>
      <li>Also maintained by W3C</li>
    </ul>
  </li>
</ul>
```

Output of Example 1:

The topics covered are:

- HyperText Markup Language (HTML)
  - A markup language based on SGML
  - Used to create web pages
  - Maintained by W3C
- Cascading Style Sheet (CSS)
  - Used to define presentation style for web pages
  - Also maintained by W3C

4.11 Tables

Table-related tags are meant for tabulating data. (Older HTML documents tend to use <table> for formatting the document to divide the document into columns/sections, which should be avoided. Use style sheet for formatting instead.)

The basic unit of a table is a *cell*. Cells are grouped into *row*. Rows are grouped to form the *table*. This corresponds well to the "row-centric" approach in the display.

The essential tags used by table are:

- <table>...</table>: contains the entire table.
- <tr>...</tr>: contains a row.

- <th>...</th> and <td>...</td>: contain a *header* cell and a *data* (*detail*) cell respectively.

Additional tags are:

- <caption>...</caption>: specifies a caption.
- <thead>...</thead>, <tbody>...</tbody>, and <tfoot>...</tfoot>: for marking out the table header, body and footer.
- <colgroup>...</colgroup> and <col>...</col>: for applying styles to column group and column respectively.

For Example:

```
<table>
  <caption>Price List</caption>
  <tr>
    <th>Fruit</th>
    <th>Price</th>
  </tr>
  <tr>
    <td>Apple</td>
    <td>$0.50</td>
  </tr>
  <tr>
    <td>Orange</td>
    <td>$0.65</td>
  </tr>
</table>
```

Price List

Fruit	Price
Apple	$0.50

Orange	$0.65

Table <table>...</table>

Function: Set up a table, consisting of rows of cells.

Three optional presentation attributes, border="*n*" (specifies the width of borders, in pixels), cellspacing="*n*" (specifies the spacing between cells, in pixels), and cellpadding="*n*" (define the spacing between the content of the cell and its boundaries, in pixels), are often used in older HTML pages but now deprecated. The now-preferred approach is to use CSS (again! but coming soon!).

Table Row <tr>...</tr>

Function: Set up a row inside a table, consisting of cells.

Table Header Cell <th>...</th>**, Table Data Cell** <td>...</td>

Function: Set up each individual cell of a row (of a table). <th>...</th> defines a header cell (usually displayed in bold with center alignment) and <td>...</td> defines a body cell.

An empty cell is typically marked as <td> </td>.

INTRODUCTION TO CSS

Why Style Sheet?

The *original* aim of HTML is to let the content providers concentrate on the contents of the document and leave the appearance to be handled by the browsers. Authors markup the document contents using markup tags (such as <p>, <h1>, <ul>, <table>, <img>) to indicate its *semantic meaning* ("This is a paragraph", "This is heading Level 1", "This is an unordered list", "This is a table", "This is an image"). The browsers then decide on how to *display* or *present* the contents in the browser's window for the web surfers.

However, HTML has gone out-of-control in the early years. Many markup tags and attributes were created for marking the appearance and the display styles (e.g., <font>, , align, color, bgcolor, link, alink, vlink are concerned about the appearance in font, color and alignment) rather than the meaning of the contents. These tags flood the document and make creation and maintenance of the contents extremely difficult. Furthermore,

over the years, we have engaged graphic designers to work on the appearance and leave the content providers to focus on the contents. Hence, there is a need to *separate the contents and presentation* of the HTML document.

The W3C (World-Wide Web Consortium @ www.w3c.org) responded to the need of separating document's contents and presentation by introducing a Style Sheet Language called CSS (Cascading Style Sheet) for presentation, and removing the presentation tags and attributes from HTML. CSS can be viewed as a *companion* of HTML. It allows web graphic designers to spice up the web pages, so that the content providers can focus on the document contents with HTML.

CSS Specifications

W3C defines three CSS levels:

1. CSS Level 1 (December 1996): CSS1 laid the ground work and introduced the selectors and most of the properties.
2. CSS Level 2 (May 1998) and CSS Level 2.1 (Last revised on June 2011): CSS2 added new features such as targeting devices and printers, and absolute positioning. CSS2.1 (@ http://www.w3.org/TR/CSS2/) touches up CSS2.
3. CSS Level 3: CSS3 is not a single piece of specification. As CSS grows, W3C decided to break it into modules, such as the Selectors module, the Values and Units Modules, the Box Alignment module, and so on. Each module is then developed independently. The CSS3 Selectors module (@ http://www.w3.org/TR/selectors/) and CSS3 Colors module (@ http://www.w3.org/TR/css3-color/) were completed in 2011. Other modules are still work-in-progress.

It is important to take note that:

- CSS is humongous!!! Most of the browsers today have yet to fully support CSS2/CSS2.1.
- CSS is a language with its own syntax, which is different from HTML and JavaScript.

What is a 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 their font properties (e.g., type face, size and weight), color properties (e.g., background and foreground

colors), alignment, margin, border, padding, and positioning. This is the same as the *styles* in any publishing software like WinWord or LaTex.

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

6. CSS By Examples

6.1 CSS Example 1: CSS Syntax and Tag-Selectors

Create the following file (using a source-code editor such as VS Code, Sublime Text, Atom, NotePad++), and save as "CSSEg1.css".

```
/* CSS Example 1 ("CSSEg1.css"): CSS Tag Selectors */
/* Rule 1: Apply to the <body> element and possibly its descendants */
body {
  font-family: "Open Sans", Helvetica, Arial, sans-serif;
  font-size: 16px;            /* 16px = 12pt */
  margin: 5px 15px 5px 15px;  /* top right bottom left - no commas in between */
  padding: 0;

  background-color: #eee;     /* light gray, same as #eeeeee */
}

/* Rule 2: Apply to <h1> to <h6> elements */
h1, h2, h3, h4, h5, h6 {
  font-family: Quicksand, "Open Sans", Helvetica, Arial, sans-serif;
  color: red;
  font-weight: bold;
  text-align: center;
}

  /* Rule 3: Specifically for <h2>. Override the previous rule */
h2 {
```

```
  color: blue;
  font-style: italic;
}
/* Rule 4: Apply to all the <p> elements */
p {
  text-align: justify;
  color: black;
}
```

A CSS style sheet provides style rules to HTML documents. You test out the above styles, by creating an HTML document, which references the CSS via the <link> element, as follows:

```
<!DOCTYPE html>
<!-- CSS Example 1. Save as "CSSEg1.html" -->
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Testing CSS Style Sheet</title>
  <link href="CSSEg1.css" rel="stylesheet">
</head>
<body>
  <h1>Test CSS Style Sheet</h1>
  <h2>This is heading level 2</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
  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.</p>
</body>
</html>
```

How it Works?

1. /* ... */ is a CSS *comment*. Comments are ignored by the browsers but provide important information to the readers, as well as the author.

2. A CSS style sheet contains *style rules*.
3. A style rule begins with a *selector*, followed by the list or presentation properties enclosed within {...}.
4. A property is identified by its name, followed by its value, separated by colon ":". The name:value pairs are separated by semicolon ";". A name could have multiple values, separated by commas "," (e.g., the font-family property).
5. A *selector* selects a set of HTML elements to apply the styles. This example uses the so-called Tag-Selector, which select all elements having the tagname. For example, the first rule is applicable to the <body>; the 3rd to <h2>, and the 4th rule to all the <p> elements.
6. Selectors having the same rules can be grouped together and separated by commas ",". For example, the 2nd rule is applicable to <h1> to <h6>.
7. Some of the properties are inherited by the nested elements. For example, the <p> nested under <body> inherits the font-family from the <body>. Some properties are not inherited.
8. If more than one rules are applicable, the properties are accumulated. But, the last rule will take effect if there is conflict. For example, both Rule 3 and 4 are applicable to <h2>. The <h2> *accumulates* the properties from both rules. It takes the font-weight:bold from Rule 3 (which is not specified in Rule 4); but uses the color:blue from the Rule 4 (instead of Rule 3).
9. Style properties:
    - The font-family list the font faces, in the order of preferences. Browsers search through the list (from the beginning) to an available font face.
    - color and background-color: specify the foreground and background colors, respectively. Color can be expressed in #rrggbb (hex value), rgb(r, g, b) (decimal value between 0 and 255), or with the pre-defined color names (such as red, blue).
    - text-align: text alignment of either left, right, justify, center.
    - font-weight: normal, bold, and others.
    - font-style: normal, italic, and others.

Take note that CSS is a language with its own syntaxes. CSS syntax is totally different from HTML!

*6.2 CSS Example 2: CSS Class-Selector, ID-Selector with and *

```
/* CSS Example 2 ("CSSEg2.css"): CSS Class and ID Selectors */
```

```
/* Rule 1: Apply to the <body> element and possibly its descendants */
body {
  font-family: "Open Sans", Helvetica, Arial, sans-serif;
  font-size: 16px;            /* 16px = 12pt */
  margin: 5px 15px 5px 15px;  /* top right bottom left - no commas in between */  padding: 0;
}

/* Rule 2: Apply to <h1> to <h6> elements */
h1, h2, h3, h4, h5, h6 {
  color: red;
  font-weight: bold;
  text-align: center;
}

/* Rule 3: Apply to the element with id="header" or id="footer" */
#header, #footer {
  background-color: #eee;  /* light gray, same as #eeeeee */
}

/* Rule 4: Apply to one unique element with id="footer" */
#footer {
  text-align: right;
}

/* Rule 5: Apply to all elements having class="new" */
.new {
  color: red;
}
 /* Rule 6: Apply to all elements having class="new" */
.highlight {
  background-color: yellow;We shall use the following HTML document to test the CSS:
```

```
<!DOCTYPE html>
<!-- CSS Example 2. Save as "CSSEg2.html" -->
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Testing CSS Style Sheet</title>
  <link href="CSSEg2.css" rel="stylesheet">
</head>
<body>
  <div id="header">
    <h1>Test CSS Style Sheet</h1>
  </div>

  <div id="content">
    <h2>This is heading level 2</h2>
    <p>Lorem ipsum dolor sit amet, <span class="new">consectetur adipiscing elit</span>, sed do eiusmod
    tempor incididunt ut labore et dolore <span class="highlight">magna aliqua</span>. Ut enim ad minim    veniam, <span class="new highlight">quis nostrud exercitation</span> ullamco laboris nisi ut aliquip ex ea
    commodo consequat.</p>
  </div>
  <div id="footer">
    <p>This page is created with <span class="new">HTML5/CSS3</span></p>
  </div>
</body>
</html>
```

How it Works?

1. In the earlier example, we use *Tag-Selectors* which select elements based on tagname. Besides the *Tag-Selector*, there are *Class-Selector* which selects elements based on class attribute; and *ID-selector* which selects an element based on the id attribute.

2. An *ID-Selector* begins with a # sign, followed by an id-name, e.g., #header and #footer, which select element with id="header" and id="footer", respectively. Since id-value is meant to be unique within an HTML document, ID-selector select at most one element.
3. A *Class-Selector* begins with a dot "." followed by a class-name, e.g., .new and .highlight, which select all elements with class="new" and class="highlight", respectively. Unlike id-value that is unique, many elements can have the same class-name.
4. In the test HTML document, we partition the body into three sections, via . We assign a unique id to each , i.e., , and to semantically identify their contents, and use the ID-Selectors #header and #footer to apply styles to the 's.
5. Similarly, we mark texts with and , and use the Class-Selectors .new and .highlight to apply styles to these texts.
6. Take note that:
    - is a block element, while is a inline element.
    - The class attribute can take multiple values, as in class="new highlight" (Line 18). Both styles are applied.

HTML id **vs** class **Attributes**

- id and class are used to reference HTML elements by JavaScript and CSS. ID can also be used as the target of HTML anchors.
- id must be unique in an HTML document, but class needs not.
- An element can have one id, but many classes.
- CSS's ID-selector begins with # (e.g., #header), Class-selector begins with dot (.) (e.g., .header).

For example,

```
<div id="header" class="highlight new big class99">.....</div>
  <!-- no other elements shall have id of "header" -->
<!-- Referenced by HTML -->
<a href="#header">Go To Header</a>
<!-- Referenced by CSS -->
<style>
#header { ..... }
.highlight { ..... }
```

```
div.new { ..... }
.new { ..... }
</style>
<!-- Referenced by JavaScript -->
<script>
var html = document.getElementById("header").innerHTML;
</script>
```

Recommendation: Use ID for JavaScript and internal <a> link (which refers to one single element in the HTML document); and class for CSS (which refers to a group of elements with the same style).

6.3 Validating CSS

You can use Online CSS Validation Services such as W3C Jigsaw (@ http://jigsaw.w3.org/css-validator) to validate your CSS file.

Try validating the above example using W3C CSS Validator.

6.4 Inspecting Element's Styles via F12 Web Developer Tools

The Web Developer Tools supports HTML, CSS, JavaScript and DOM. You could select (inspect) an HTML element, and it will show you all the CSS rules that are applied to that elements from all the sources (inline, embedded, external), and how the rules were merged (calculated) and conflicts were resolved. You can temporarily disable a rule, and edit a rule to check the effect instantly.

To debug CSS:

1. F12 to launch Web Developer Tools.
2. To inspect the style of an HTML element ⇒ Choose the "HTML" panel ⇒ Click on the "Inspect" button and then select the HTML element of interest from the browser window ⇒ You can check/modify the "Style", "Layout", "DOM" and "Events" (on the right panel) associated with the selected element.
3. To check/modify the CSS Style rules ⇒ Choose the "CSS" panel.

7. CSS Basics

7.1 CSS Syntax

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

The syntactic rules are:

1. A *style rule* consists of a *selector* which selects the HTML elements it operates upon, and a list of style property name:value pairs enclosed in braces {...}, as follows:

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

   For example,

```
body {   /* Apply to <body> and possibly its descendants */
  font-family: "Open Sans", Helvetica, Arial, sans-serif;
  font-size: 16px;
  margin: 10px auto;  /* top-down right-left */
  padding: 0;
}
```

   This *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 its descendants, unless they are overridden by other style rules.

6. The name:value pairs are separated by semicolon ";". You can omit the last semi-colon before the closing brace "}". But I recommend that you keep it, so that it is easier to include new entries without a missing ";".
7. The name and value are separated by a colon ":" in the form of name:value.
8. Multiple values for the same property name are separated by commas "," (as in the font-family). However, multiple parts of the same property value are separated by space " " (as in the margin, which has a value with 4 parts).

9. Values containing space must be quoted, e.g., "Times New Roman" or 'Times New Roman'.
10. Extra whitespaces (blank, tab and newline) are ignored.
11. If the same set of styles is applicable to more than one elements, the selectors can be grouped together in one single rule (called Group-Selector). The tagnames are separated by commas ",". For example, the following rule apply to elements <h1> to <h6>:

```
h1, h2, h3, h4, h5, h6 {
  text-align: center;
  font-family: Quicksand, "Open Sans", Helvetica, Arial, sans-serif;
}
```

15. Comments can be inserted inside the style sheet enclosed between /* and */. The end-of-line comment // is not applicable in CSS?!

CSS Syntax vs. HTML Syntax

CSS and HTML have different syntaxes!!! For example, HTML's attributes uses "=" to separate the name and value, in the form of name="value"; the name-value pairs are separated by spaces, as follows:

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

7.2 Inline, Internal and External Styles

There are three places where you can define style rules:

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

Inline Styles

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

```
<!DOCTYPE html>
```

```
<html>
<body>
  <p style="font-size:16px; font-family:monospace">This paragraph uses 16px monospace font.
</p>
  <p>This paragraph uses default font.</p>
  <p>This paragraph uses <span style="font-size:12px">12px inside this span</span>
  but default font size here.</p>
</body>
</html>
```

This paragraph uses 16px monospace font.

This paragraph uses default font.

This paragraph uses 12px inside this span but default font size here.

Take note that the name and value are separated by colon ':', and the name:value pair are separated by semicolon ';', as specified in the CSS syntax.

The scope of an inline style is limited to that particular element. 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.

Internal (Embedded) Styles

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

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      background-color:cyan;
    }
    h2 {
      color:white;
      background-color:black;
```

```
    }
    p.monospace {
      font-size:16px;
      font-family:monospace;
    }
    p.f20px {
      font-size:20px;
    }
  </style>
</head>
<body>
  <h2>H2 is white on black</h2>
  <p>This paragraph is normal.</p>
  <p class="monospace">This paragraph uses 16-px monospace 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!?

NOTE: HTML4/XHTML1 require an additional attribute type="text/css" in <style> opening tag.

External Styles

The style rules are defined in an external file, and referenced in an HTML document via the <link> element 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> element in the HEAD section:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <link href="TestExternal.css" rel="stylesheet">
</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>
```

You can use multiple <link> elements to include multiple CSS files.

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

NOTE: HTML4/XHTML1 require an additional attribute type="text/css" in <link> element.

Linking to External Style Sheet using CSS @import **Directive**

Besides the HTML <link> element, you can also use CSS's @import directive to link to an external style sheet under the HTML <style> tags, as follows:

```
<!-- in HTML file -->
<style>
  @import url("cssURL1.css");
  @import url("cssURL2.css");
</style>
```

@import is a CSS directive (part of CSS language). It can also be used in one CSS file to include rules from another CSS file, for example,

```
/* in CSS fie */
@import '/css/more-styles.css';

/* Import an external Google font */
@import 'https://fonts.googleapis.com/css?family=Open+Sans';

/* Second argument is media queries on device and feature */
@import '/css/print-styles.css' print;
@import '/css/landscape.css' screen and (orientation:landscape);
```

Priority

Inline styles have the highest priority, followed by internal styles, and followed by external styles.

7.3 Inheritance

Many (but not all) CSS properties, such as color and font-family, affect not only the elements selected by the selector, but also *inherited* by their descendants.

Inheritance is a big *time-saver* for designing styles. For example, you set the default color and font-family in the <body> element, which will then be inherited by all the elements. You then override the default properties for specific elements, if needed.

Some properties (such as border, margin, padding, width, height, background-color) are not inherited. This is for good reason. For example, if border is defined for <ul> and is inherited, then all descendants (<li>) will be drawn with border!

Special Property Value: inherit

You can use a special property-value called "inherit" to inherit the property from its ancestor. 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 the first is nested under the <p> tag, the border property is not inherited from the ancestor. 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 in the Class Selector .inherit-border.

7.4 Resolving Conflicting Rules

Style Conflict

Style conflict on an element arises:

1. A property is inherited from multiple ancestors.
2. More than one rules are applicable to the element. For example, Tag-selector p, Class-selector .red and ID-selector #comment are all applicable to element <p id="comment" class="red">.

Nearest Ancestor Wins

If a property is not defined for an element and is inheritable, it will be inherited from the nearest ancestor.

Specificity

Specificity means that "the more specific the selector, the stronger the rule". For example,

```
<!DOCTYPE html>
<html>
<head>
<style>
p     { color:black; background-color:white; }
/* Override the color properties */
p.red { color:red;   }
p#id1 { color:yellow; background-color:lightblue; }
p#id2 { color:blue;  }
p#id1 { color:green; }
</style>
</head>
<body>
<p id="id1">Paragraph with id of "id1" (green)</p>
<p id="id2">Paragraph with id of "id2" (blue)</p>
```

```
<p class="red">Paragraph of class of "red" (red)</p>
<p id="id1" class="red">Paragraph with id of "id1" and class of "red" (green)</p>
<p>Paragraph without id and class with default colors (black)</p>
</body>
</html>
```

The p Tag-selector is the most general, which selects all the <p> elements; the p.red Class-selector selects a class of <p> elements with attribute class="red"; the p#id1 and p#id2 ID-selectors select only one element each with a unique id value. The ID-selector is the most specific (and takes precedence); followed by Class-selector; and followed by the general Tag-selector.

Locality

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 ID-selector for id1, the latter takes effect.

The inline style (applied to a specific tag via style attribute) overrides the internal style (defined in <style>) and external style sheet (defined via <link>). For internal and external styles, the order of <link> and <style> elements determine the precedence. It is recommended to place the <link> before <style> so that the internal styles can override the external styles.

Preventing Override: !important

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

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

7.5 How to Use CSS for Styling HTML Document?

To use CSS to style your website for good and consistent look and feel, you need to properly structure and partition your web pages.

HTML Division **and Span** **Elements**

Prior to HTML5, two HTML elements, 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, highlight text, and so

on). is a *block element* which is rectangular in shape; while is an *inline element* which spans a sequence of characters.

The and are *generic* tags for identifying contents. They do not possess any inherent visual properties (unlike <h1>, <p>, which are expected to be presented in a certain way). They shall be further qualified with id or class attribute, and attached with CSS styles selected via the ID-selector or Class-selector.

Modern-day HTML pages use and *extensively* to structure the document for applying CSS styles. Old HTML pages uses tables and frames, which should be avoided. HTML5 introduces new semantic elements such as <header>, <footer>, <section>, <nav>, <article> to help you better structure your page.

HTML Attributes id **and** class

All the HTML elements supports two optional attributes: id="*id-value*" and class="*class-value*".

- You can assign an id="*id-value*" attribute 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 can be used by CSS (as well as JavaScript) to select that particular 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>
```

- Similarly, you can assign class="*class-value*" attribute to a class of elements having the same presentation properties and appearance. 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* (hence, the keyword *class*). The class attribute is primarily used by CSS to apply a common set of styles to all the elements of the same class. For example,

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

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

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

CSS Tag-Selector (T**)**

A CSS *Tag-selector* selects HTML elements based on the *tag-name*, e.g.,

```
/* tag selector */
```

```
h2 { background-color:black; color:white; text-align:center; }
```

CSS ID-Selector (#D)

A CSS *ID-selector*, which begins with a '#' followed by an id value, selects a unique element (because id value is supposed to be unique) in the document. For example,

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

CSS Class-Selector (.C)

A CSS *Class-selector*, which begins with a '.' followed by a classname, selects ALL elements having that class value. For example,

```
/* Class-selector (class value needs not be unique) */
.highlight { background-color: #ff0; }
.underline { text-decoration: underline; }
.green     { color:green; text-decoration:underline; }
.blue      { color:blue;  }
```

Applying CSS

1. Firstly, partition your web page in semantic partitions using the and elements, or the newer HTML5's <header>, <footer>, <section>, <article>, <nav> elements.
2. Assign id or class to each of the partitions. Use id if it is unique (in formatting); otherwise, use class (more than one partitions have the same formatting).
3. Write the CSS style rules for tag, id and class.

Example: The following HTML page is divided into three sections. Selected texts are marked with the tags, with class of "green" and "blue".

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Structure Web Page and apply style</title>
  <meta charset="utf-8">
  <link href="MyStyle.css" rel="stylesheet">
```

```
</head>
 <body>
<header>
  <h1>Heading</h1>
</header>
 <section id="content">
<h2>Hello</h2>
<p>Lorem ipsum dolor sit amet, <span class="green">consectetur adipisicing</span> elit, sed d
o
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 consequ
at.
Duis aute irure dolor in</span> reprehenderit in voluptate <span class="blue">velit esse</span
>
cillum dolore eu fugiat nulla pariatur.</p>
</section>
 <footer>
  <p>Footer</p>
</footer>
</body>
</html>
```

The CSS file "MyStyle.css":

```
[TODO]
```

8.2 CSS Length Measurements

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

```
html {
  font-size: 16px;    /* base measurement for rem (CSS3) */
}
p {
  font-size: 1rem;    /* relative to root html element */
```

```
  width: 80%;         /* 80% of the parent's width */
  margin: 0.5em 2em;  /* relative to current font-size */
  border: 5mm;        /* absolute millimeters */
  padding: 0;
  line-height: 140%;  /* 1.4 times of the current font-size */
}
```

There are two types of length measurements: *relative* (to another length property) 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.
- px: px is a measurement unit created for the CSS, where the thinest line shall have width of 1px. Today, it is defined as 1px=1/96 inch. Since 1pt is 1/72 inch; 12pt = 16px = 1/6 inch = 0.42cm.

The relative units are:

- % (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 font-size of <p> to 1.2 times of the parent (possibly a or <body>).
- rem (CSS3): relative to the font-size of the root or <html> element.
- vw, vh (CSS3): one percent of viewport width and height respectively.

- vmin, vmax (CSS3): one percent of the viewport smaller dimension or larger dimension respectively, i.e., 1vmin is equal to the smaller of 1vh or 1vw; 1vmin is the larger of 1vh or 1vw.
- 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,

```
h6 {
  font-size: 1.2rem;    /* 1.2 times of the <html> font-size */
  width: 80%;           /* 80% of the parent's width */
  margin: 0.5em 1.2em;  /* relative to the current font's letter 'm' */
  padding: 10px;        /* 10 logical 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 */
```

NOTE: In HTML tag attributes, such as width="400", the bare number is measured in pixels.

Recommendation

- Define a font-size for <html>.
- Use rem for font-size of other elements - relative to <html>.
- Use em for other properties such as margin and padding, which is relative to the current font-size.
- Use other relative measurement such as % if appropriate. Avoid absolute measurements.

For example,

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
<meta charset="utf-8">
<title>Measurement Test</title>
<style>
html {
  font-size: 16px;
}
input[type="button"] {
  font-size: 1rem;
  padding: 0.4em 2.5em;
}
input[type="text"] {
  font-size: 1rem;
  padding: 0.4em;  /* same top/bottom as other input elements for proper alignment */
}
input[type="number"] {
  font-size: 1rem;
  padding: 0.4em;  /* same top/bottom as other input elements for proper alignment */
}
</style>
</head>
<body>
<input type="button" value="Button">
<input type="text" value="This is a Text Box">
<input type="number" value="5" min="0" max="10" size="2">
</body>
</html>
```

8.3 CSS Box Model and Properties 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 always *rectangular* in shape and exhibits the so-called *box model*, with *four virtual rectangles* wrap around its "*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.

Inspecting the Box Model via Web Developer Tools

CSS width **and** height **Dimension Properties**

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

- width|height: auto|*n*|*n*%: The width and height are specified in units such as px (pixels), or *percent* (*relative to the parent element*).
- max-width|max-height|min-width|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|margin-right|margin-bottom|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.
- (Shorthand) margin: *top right bottom lef*t
  (Shorthand) margin: *top right-left bottom*
  (Shorthand) margin: *top-bottom right-left*
  (Shorthand) margin: *all*
  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.
  Take note that there is no commas between the items, as all items are considered to be one property value.
- padding-top|padding-right|padding-bottom|padding-left: *n*|*n*%: Set the four paddings individually, similar to margin-*xxx*.

- (Shorthand) padding: *top right bottom left*
  (Shorthand) padding: *top left-right bottom*
  (Shorthand) padding: *top-bottom left-right*
  (Shorthand) padding: *all*
  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-style-right, 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-color-right, border-color-bottom, border-color-right to set the four borders individually..
- (Shorthand) border: *width style color*
  Shorthand notation to set all the properties of the borders, in the order shown. You can also use shorthands 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.

Example: [TODO]

More Margin Properties

- margin: 0 auto: set top and bottom margins to 0. Setting left and right margins to auto equally distribute the extra horizontal space, and hence, center the element horizontally.
- Collapsed Margins: When margin of two elements touching each other vertically, they collapsed (with the larger margin). But when two margins touches horizontally, they do not collapse.
- Negative Margin: margin is one of a few CSS properties that can be set to negative. This is used to overlap elements (without using absolute positioning). Padding and border cannot be negative.

More Border Properties

- border-radius: can be used to create a rounded border. You can also use it to convert a box to a circle (with border-radius: 50%).
- border-image: can be used to create a multi-color border.

Example [TODO]

CSS outline **Properties**

In addition to border, outline is a line that goes around the element, outside of the border, which does not take any space in the box model, and does not affect the position of the element.

- outline-color:
- outline-style: dotted|dashed|solid|double|groove|ridge|inset|outset: similar to border.
- outline-width:

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.

Example: [TODO]

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.

Example:

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;
 }
```

8.4 CSS 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|xx-large|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 small-caps is smaller than the uppercase.
- (Shorthand) font: *style variant weight size*/*line-height 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: 16px;
  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.

Example: [TODO]

Google Fonts

"Google Fonts" @ https://fonts.google.com/ is a huge set of high-quality, free and open-source fonts. It is certainly the choice of the fonts today.

To use particular fonts, e.g., "Roboto" and "Roboto Mono", you can either:

1. Add a <link> to CSS (in your HTML's HEAD section, e.g.,

```
<link rel="stylesheet" href='https://fonts.googleapis.com/css2?family=Open+Sans&family=Roboto+Mono&display=swap'>
```

2. Use a @import in your CSS (you need not update all your HTML pages):

```
@import url('https://fonts.googleapis.com/css2?family=Open+Sans&family=Roboto+Mono&display=swap');
```

8.5 CSS 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.

8.6 CSS 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.
- (Shorthand) background: *color image repeat attachment position*: one-line shorthand notation.

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.

8.7 CSS list-style Properties

The properties are:

- list-style-type: none|disc|circle|square
  list-style-type: decimal|decimal-leading-zero
  list-style-type: lower-alpha|upper-alpha|lower-roman|upper-roman|lower-greek|lower-latin|upper-latin
  Set the style of the list item <li> marker (bullet) for <ul>, <ol> respectively.
- 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.
- (Shorthand) list-style: *type position image*: Shorthand notation to specify all the properties of the list.

8.8 CSS Table Properties

- 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