HTML – Hypertext Markup Language

DM2113 Web Development – Lesson 2

In this lesson

- Introduction to HTML
- HTML5 and Standard-Compliant Mark-up
- HTML BASICS
- HTML Tags: Special Characters, Font, Colour, List, Tables & Cells, Hyperlinks and Images

INTRODUCTION TO HTML

HTML is made up of elements (often called tags) that build the contents of a web page. The differences between HTML and other programming languages include:

- HTML is not compiled. It is a text file which a browser interprets it.
- HTML is readable. Unlike programming languages which you have to learn the language to really understand it, we can at least guess what an <img> tag does.
- The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

World Wide Web Consortium (W3C) standardisation

The World Wide Web Consortium (W3C), founded in 1994 by Tim Berners, is an international consortium where Member organizations, a full-time staff, and the public work together to develop Web standards.

W3C's mission is: To lead the World Wide Web to its full potential by developing protocols and guidelines that ensures long-term growth for the Web.

A W3C Recommendation means that the specification is stable, that it has been reviewed by the W3C membership, and that the specification is now a Web standard.

Version history of the standard (http://www.w3.org/)

- HTML 2.0 — September 22, 1995,
- HTML 3.0 — proposed by W3C in March, 1995
- HTML 3.2 — January 14, 1997,
- HTML 4.0 — December 18, 1997, support presentational language, cascading style sheets.
- HTML 4.01 — December 24, 1999, recommended by the W3C
- XHTML 1.0 — January 26, 2000. W3C introduced XHTML to succeed HTML.
- HTML 5 —2004 Web Hypertext Application Technology Working Group (WHATWG).
    - In 2006. W3C and WHATWG work together on the development of HTML5

WHAT IS HTML5?

HTML5 will be the new standard for HTML.

HTML5 is collaboration between the *World Wide Web Consortium (W3C)* and the *Web Hypertext Application Technology Working Group (WHATWG)*.

HTML5 was designed to replace HTML4 and XHTML, and the HTML DOM Level 2.

HTML5 is designed to work on PC, Tablet, Smartphone, or Smart TV.

Some rules for HTML5 were established:

- New features should be based on HTML, CSS, DOM, and JavaScript
- Reduce the need for external plugins (like Flash)
- Better error handling
- More mark-up to replace scripting
- HTML5 should be device independent
- The development process should be visible to the public

Today's market consists of different browser technologies, some browsers run Internet on computers, and some browsers run Internet on mobile phones and potable-devices that do not have the resources or power to interpret a "bad" mark-up language.

Web Interoperability

There are many different platforms, operating systems, and browsers available. Products are competing with each other, and to outdo the competitors, developers are always trying to provide something different (additional features).

This diversity results in different operations and functionalities. *What would happen if these developers do not have proper application of the latest standards?*

Web interoperability ensures that standard-compliant web pages can be viewed in any browser under various operating systems, from Windows to Mac OS and Linux, and not only on desktop computers but also on mobile devices, including tablet PCs and smartphones.

Several technologies support interoperability and should be used in web development, for example, UTF-8 character encoding, XML documents, structural and semantic mark-up with XHTML or HTML5 [113], DOM scripting, ECMAScript, CSS-based layout, separated structure, presentation and behaviour, equations described in MathML, and semantic metadata.

Some standards:

- Mark-up languages such as HTML and XHTML are good examples for device independence standards.
- CSS can be used to provide device independence through additional style sheets for devices.
- Java applets can be executed on a variety of devices under different platforms, because Java is a cross platform programming language.
- Image file formats such as JPEG, TIFF, or GIF are also device independent files. In document publishing and sharing, PDF is a classic example for device independence.

The Problem of Non-standardised Documents

While we applied proper coding standards in programming, most web developers do not take mark-up practices seriously.

Many web designers focus on the visuals and are not really interested in mark-up or style sheets. Company leaders focus mainly on the content. They do not realise that standard compliance could be the solution for many of their problems, such as browser-dependent web pages, incorrect rendering, or poor functionality.

The major drawbacks of non-standardised documents are the following:

- Bad mark-up significantly increase code length, complexity, download, and rendering time.
- Incorrect rendering.
- Low level of web accessibility. Web may not be usable by people of all abilities and disabilities.
- Low level of backward compatibility.
- Difficult updating and maintenance.

Standard-Compliant Mark-up

Valid, standard-compliant mark-up has several advantages.

- Search engine crawlers can index documents more adequately, and the content is basically search engine optimised.
- Optimal content lengths and file size, as well as optimal storage
- Easier to maintain and update than the mark-up that violates standards.
- Compatibility with current and future browsers is guaranteed.

To reach a certain level of quality assurance and achieve basic requirements for standard compliance, proper development practices should be observed. These practices provide adequate and more expectable functionality and behaviour, usability, and stability, as well as faster downloading and rendering.

HTML Validator

http://validator.w3.org/

Use the validator to checks the mark-up validity of you Web documents.

Common HTML Mark-up Mistakes to Avoid!

Below are some of the mistake and ways to avoid them

No Doctype

The Doctype describes what kind of HTML you are using.

Use of Uppercase (instead of lowercase) for Tags and Attributes

HTML 5 is not case-sensitive. The convention is to use all lowercase for tags and attributes.

Improperly Nesting Tags

HTML tags must be closed in the opposite order than which they were opened.

```
<p>This is a paragraph with <strong>bold text!</p></strong>

<p>This is a paragraph with <strong>bold text!</strong></p>
```

Place Block Elements Inside Inline Elements

Block elements (e.g. divs and paragraphs) make up the structure of the document. Inline elements (e.g. anchor and span tags) reside in these blocks.

```
<a href="#"><h2>Block inside Inline</h2></a><!-- wrong! -->

<h2><a href="#">Inline inside Block</a></h2> <!-- correct -->
```

Forget to Close an Element

Most HTML elements have a separate closing tag (e.g. , </p>, but other elements (e.g. input, img and meta) are self-closing

Forget To Open and Close Quotes

All attributes, even numerical ones, must be quoted.

```
<img src=smiley.gif alt=Smiley face> <!-- wrong! -->

<img src="smiley.gif" alt="Smiley face"> <!-- correct! -->
```

Use Physical Style Tags (instead of Logical Style Tags)

Physical Style Tags (e.g. <b> and <i>) should be avoided. Logical Style Tag (and) should be used.

Forget to Convert Special Characters

Any "&" signs must be converted to the HTML entity &

Same apply to "Unencoded characters" in URLs.

Forget to Add ALT Attribute to Image Tags

ALT attribute describes the context of the image. If the image is just for show, an empty ALT attribute should be used.

Use Inline Styles

Document structure and styling should be separated. Don't place styling directly in the HTML document. (More details in CSS)

MY FIRST WEB PAGE

Let begin by putting together a web page,

1. Start Notepad++ and enter the following codes,

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>My First Web Page</title>
  </head>
  <body>
    This is my first attempt at creating a Web page.
  </body>
</html>
```

`myFirstWeb.html`

2. Save the file as myFirstWeb.html.
3. Open a browser (e.g. Internet Explorer) and open the file.
4. Congratulation, you have created your first web page.

Workshop Explained

Document Type Declaration <!doctype>

All *HTML5* documents should begin with

```
<!doctype html>
```

It tells the browser what form the page will take. The browser must render the page in accordance with the specifications given.

The <!doctype> is simpler than ever, no need for excessive body attributes.

The <html> element

The *<html>...</html>* surround all HTML coding in the document. The enclosed information contains HTML elements and should be rendered as such in the browser.

Optional. In conformance with *HTML5* standards, the opening tag includes a reference to the location namespace of the validation standards to be applied to this document, along with attributes that specify the language used (English in this case):

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
```

The <head> element

The *<head>...</head>* encloses the head section of the HTML document. The head section supplies a *title* for the document along with other information related to formatting and indexing of the document.

The <title> element

The *<title>...</title>* gives a title to the document.

```
<title>My First HTML Web Page</title>
```

This tag encloses the string "My First HTML Web Page" that appears in the browser's Title Bar when the page is opened. The <*title*> tag is *optional* but provides helpful identification information to the person visiting the various pages of your Web site.

The <meta> element

Metadata is data (information) about data. The <*meta*> provides metadata about the HTML document. This element does not have any content and *will not be displayed* on the page. However, metadata are stored by the browsers. There are several kinds of *<meta>* tags, the most important is the *charset* that identifies the character set that the file will be using and it should be included in all page structures.

```
<meta charset="utf-8" />
```

This line tells the browser what kind of text will be used, including the actual character set.

Other <meta> tags include

```
<!-- Example 1 - Define keywords for search engines: -->
<meta name="keywords" content="HTML, CSS, XML, JavaScript">

<!-- Example 2 - Define a description of your web page: -->
<meta name="description" content="Free Web tutorials" />

<!-- Example 3 - Define the author of a page: -->
<meta name="author" content="Mr Quah" />

<!-- Example 4 - Refresh document every 30 seconds: -->
<meta http-equiv="refresh" content="30" />
```

The <body> Element

The bulk of the coding of a HTML document appears in the body section surrounded by the *<body>...</body>*. Only information appearing inside this tag is displayed in the browser window. In its simplest form, the body section contains plain text that is displayed in the default font style inside the browser window.

HTML BASICS

HTML Elements & Attributes

Every HTML document is mark-up of a hierarchical structure of elements and their content. Generally, an element consists of an *opening (or start)* tag, some *content*, and a *closing (or end)* tag. *Some elements have empty content and do not have closing tag*.

Tags are labels used to surround the content to be formatted and are enclosed inside angled brackets *("<" and ">")* to identify them as mark-up instructions. The browser displays the page in the specified layout structure and style according to the elements.

Attributes provide additional information about HTML elements. The attributes are always specified in the opening tag. We will discuss about this later.

HTMLBasics.html

Headings

Headings are defined with the *<h1> to <h6>* tags. <h1> defines the largest heading. <h6> defines the smallest heading.

```
<h1>This is the biggest heading used for main headings</h1>
<h2>This is a bigger heading</h2>
<h3>This is a big heading</h3>
<h4>This is a small heading</h4>
<h5>This is a smaller heading</h5>
<h6>This is a smallest heading</h6>
```

Headings are block elements and automatically add *an extra blank line before and after*.

Use HTML headings for headings and not to make text BIG or bold. Search engines use your headings to index the structure and content of your web pages.

Paragraphs

Paragraphs are defined with the *<p>* tag.

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

Paragraphs are block elements and automatically add *an extra blank line before and after*.

Line Breaks

The *
* tag is used when you want to end a line, but don't want to start a new paragraph. The *
* tag forces a line break wherever you place it.

```
<p>This <br /> is a para<br />graph with line breaks</p>
```

The *
* tag is an empty tag. *It has no closing tag*.

HTML Lines

The <hr />tag creates a horizontal line in an HTML page and can be used to separate content.

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

Like the line breaks, it does not have a *closing tag*.

Comments in HTML

The comment *<!-- -->* tag is used to insert a comment in the HTML source code. *A comment will be ignored by the browser*. You can use comments to explain your code, which can help you when you edit the source code at a later date.

```
<!-- This is a comment -->
```

Note that you need an exclamation point after the opening bracket, but not before the closing bracket.

TEXT FORMATTING

textformat.html

Physical Style Tags

Physical style tags enclose the string of characters to be displayed in the specified style (e.g. bold and italic). These tags are common to all browsers and are displayed the same in all browsers. *We will not use this*.

Logical Style Tags

Logical style tags are inline elements. They *do not have standardized meanings* in all browsers, and there is no requisite on how they should be displayed. However, they are more broadly applicable than physical style tags in permitting both visual and non-visual rendering of styles.

For people who are visually impaired, for example, the *<b>* physical style tag may be meaningless since it renders text visually in bold characters. For example, by using the visually comparable ** logical style tag, a person using special reader software hears the text with audio emphasis (e.g. louder).

Logical style tags are preferred over physical style tags to meet these sorts of browsing contingencies.

Physical or Logical?

SPECIAL CHARACTERS

SpecialCharacters.html

There are certain text characters that cannot be displayed in the browser by typing them directly into the text editor. Some of these characters have special meaning in HTML and, rather than displaying them, the browser interprets the characters as HTML code.

- *As < and > are used to define tags. To display <p> on the browser, we write the codes <p >.*
- These characters are prefixed with an ampersand (*&*) and suffixed with a semicolor(*;*) to identify them as special characters.
- *Symbols*, such as © (copyright) and ™ (trademark), do not have keyboard equivalents. Still, you need a way to represent them on the Web page.

IMAGES

Images.html

Images are embedded into the web page using the *<img>* tag. Image tags are inline elements.

```
<img src="same5001.jpg" width="400" height="400" alt="2005 Year
of the Rooster $10 Silver Colour Piedfort Proof Coin" />
```

<img /> element does not have content and closing tag, it contains attributes only.

src Attribute

To display an image on the page, you need to identify the source (or src) or *the URL* of the image you want to display.

In the above example, the URL of the image is "*same5001.jpg*". It means that the image file (same5001.jpg) is stored in the same directory as the HTML file (images.html).

Linking to file/directory

For image files (or documents) in a different directory, there are three kinds of pathnames that can be used in the scr:

- *Relative Pathname* - Points to files based on their locations *relative to the current file*,
  - eg. "images/5001.jpg"
  - ".." means parent directory.
- *Remote Pathname* - Specifies the *web address* of a file,
  - eg. "http://172.21.128.133/DM2113/examples/images/5001.jpg"
- *Absolute Pathnames* - Points to files based on their *absolute locations* on the file system,
  - eg. "/DM2113/examples/images/5001.jpg "

In the above examples, the three sources identify the same image file (e.g. 5001.jpg) on the server. However, relative pathname is preferred over the other method because of ease of portability. The root of the website can change without changing the pathname.

Try it!

alt Attribute

The *alt* attribute specifies an alternate text for an image.

The alternative text is display if the image cannot be found at the specified URL or the mouse is over the image.

The primary purpose of the alternative text is to describe the image for the benefit of the visually-impaired, or for users who have image display disabled in their browser.

height and *width* attributes

The height and width attributes are used to specify the height and width of an image

It is a good practice to specify both the height and width attributes for an image. If these attributes are set, the space required for the image is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the image. The effect will be that the page layout will change during loading (while the images load).

LINKS

Links (or Hyperlinks) are found in nearly all Web pages. Links allow users to click their way from page to page.

Without the links, the World Wide Web would be utterly useless. Links form the basis of the web because they are the means by which a user can navigate from page to page or from site to site.

To create a link in HTML, you need two things:-

- The name of the file (or the URL of the file) to which you want to link.
- The "hotspot" (a word, group of words, or image) identifies the hyperlink on the page.

Here is an example of a link, in this case specifying a URL:-

```
<a href="http://www.nyp.edu.sg/">Nanyang Polytechnic</a>
```

- The *<a>* tag is used to create anchors for links. (*a* is short for "anchor"). The is used to close the tag.
- *href* is short for "hypertext reference" which is the URL
- Then comes the *link text* (or content); the text that the browser will display as a link (e.g. Nanyang Polytechnic) instead of the actual URL.

The <a> tags are inline elements.

By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

On-Page Links (The id Attribute)

Links are normally made between different Web documents so that visitors can navigate between pages. Links can also be made to different locations in the same document.

In order to create *on-page links* you need to code the pair of <a> anchor tags shown below.

The *id attribute* is used to create a bookmark inside an HTML document.

```
<a href="#name">link text</a>
<a id="name">target text</a>
```

Even though the destination text strings are enclosed in *<a>* tags they are *not coloured or underlined as are normal links*. They are "invisible" targets for links (using name attributes), not links themselves (using href attributes).

Link to a target on another page

To create a link to the "ITEM1" from another page:

```
<!-- from the same directory to this page -->
 <a href="links.html#ITEM1">Go to Item 1</a>
 <!-- using Absolute Pathname  -->
 <a href="http://172.21.128.133/DM2113/examples/links.html#ITEM1"
>Go to Item 1</a>
```

LISTS

lists.html

There are three types of lists

- Ordered lists (numbered items)
- Unordered lists (bulleted items)
- Definition Lists (terms and definitions) – *Not common*

The first two lists are commonly used. The latter list is similar in display to a series of *block quoted paragraphs*.

All tags are block elements.

The Ordered List

Ordered list opens with the *
* tag and closes with the *
* tag.

Each item of the list also has its own opening and closing tag. *
* opens the item and *
* closes it.

```
<ol>
  <li>First list item</li>
  <li>Second list item</li>
  <li>Third list item</li>
  <li>Fourth list item</li>
</ol>
```

The start Attribute

When using decimal numbers for an ordered list you can choose the beginning sequence number by coding the optional *start="n"* attribute for the *<ol>* tag.

```
<p>This is the beginning of the list:</p>
<ol>
  <li>List Item A</li>
  <li>List Item B</li>
</ol>
<p>This is a continuation of the list:</p>
<ol start="3">
  <li>List Item C</li>
  <li>List Item D</li>
</ol>
```

The Unordered List

This type of list opens with the <ul> tag and closes with the tag. List items use the tags as ordered lists.

```
<ul>
  <li>First list item</li>
  <li>Second list item</li>
  <li>Third list item</li>
  <li>Fourth list item</li>
</ul>
```

Nested Lists

In addition to putting line breaks, images, links, etc. inside a list item, you can put other lists. Lists inside a list item are called nested lists.

```
<ul>
  <li>First list item</li>
  <li>Second list item
    <ul>
      <li>First nested item</li>
      <li>Second nested item</li>
      <li>Third nested item</li>
    </ul>
  </li>
  <li>Third list item</li>
<li>Fourth list item</li>
<ul>
```

Definition lists

A definition list is a series of terms and definitions offset from surrounding text by blank lines. The terms in the list are blocked at the left margin; definitions are indented and word wrapped on the following lines.

Each list item has two parts.

- A term
- A term's definition

Each part of the glossary has its own tag.

- <dl> and </dl> are used to open and close the list.
- <dt> and </dt> open and close the "term" part and
- <dd> and </dd> are used to open and close the "definition" part.

```
<dl>
  <dt>James T. Kirk</dt>
  <dd>The heroic captain of the USS Enterprise has distinguished
himself countless times when facing the unknown.</dd>
  <dt>Spock</dt>
  <dd>Kirk's trusted science officer can always be depended upon
for solutions to difficult problems as well as unswerving loyalty
to his commander.</dd>
  <dt>Montgomery Scott</dt>
  <dd>The most talented engineer in Starfleet, Scotty has used
his skills to save the USS Enterprise and keep her
shipshape.</dd>
</dl>
```

TABLES AND CELLS

Tables.html

Tables serve two primary functions in Web page design:

- *Presentation of information*. General display of data in rows and columns. It helps the viewer sort through masses of data to understand their underlying structure and content.
- *Management of document layout*. It can be used to structure the layout of a Web page regardless of its content.

Table Element

Tables are defined with the *<table>* tag.

A table is divided into *rows* (with the *<tr>* tag), and each row is divided into *header cells* (with the *<th>* tag) or *data cells* (with the *<td>* tag).

- The letters td stands for "table data," which is the content of a data cell.
- A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

```
<table>
   <th>
      <td>Header 1</td>
      <td>Header 2</td>
   </th>
   <tr>
      <td>row 1, cell 1</td>
      <td>row 1, cell 2</td>
   </tr>
   <tr>
      <td>row 2, cell 1</td>
      <td>row 2, cell 2</td>
   </tr>
</table>
```

A simple 2x2 Table

Table tags are block elements.

By default, *<table> tags produce no borders around the table or around its cells*. If you wish to display table borders, *style sheets* must be applied to the table and to its cells to display borders around both.

```
<style>
  table {border:outset 1px}
  td    {border:inset 1px}
</style>
```

The entire table is surrounded by a 1-pixel outset border and each of the cells is surrounded by a 1-pixel inset border.

Table Size

By default, *the size of a table depends on the size of the data* appearing in its cells. A column is as wide as the widest entry in the column; all rows are as wide as the combined width of data within the widest cells.

Data are aligned horizontally left and vertically centred within the cells.

Nested Tables

Tables can be nested; that is, tables can appear within cells of a table. The cell expands in size to permit full display of the nested table.

```
<table>
<tr>
  <td>Cell 1</td>
  <td>Cell 2</td>
</tr>
<tr>
  <td>Cell 3</td>
  <td>
    <table>
    <tr>
      <td>Cell A</td>
      <td>Cell B</td>
    </tr>
    <tr>
      <td>Cell C</td>
      <td>Cell D</td>
    </tr>
    </table>
  </td>
</tr>
</table>
```

Table Headings

Headings in a table are defined with the *<th>* tag.

```
<table>
   <tr>
      <th>Heading</th>
      <th>Another Heading</th>
   </tr>
   <tr>
      <td>row 1, cell 1</td>
      <td>row 1, cell 2</td>
   </tr>
   <tr>
      <td>row 2, cell 1</td>
      <td>row 2, cell 2</td>
   </tr>
</table>
```

Empty Cells in a Table

Table cells with no content are not displayed very well in most browsers.

```
<table>
   <tr>
      <td>row 1, cell 1</td>
      <td>row 1, cell 2</td>
   </tr>
   <tr>
      <td>row 2, cell 1</td>
      <td></td>
   </tr>
</table>
```

Note that the borders around the empty table cell are missing.

To avoid this, add a *non-breaking space ( )* to empty data cells, to make the borders visible:

```
<table>
   <tr>
      <td>row 1, cell 1</td>
      <td>row 1, cell 2</td>
   </tr>
   <tr>
      <td>row 2, cell 1</td>
      <td> </td>
   </tr>
</table>
```

Formatting Tables

formatTables.html

colspan and rowspan

The way to vary the number of cells in the columns or rows of a single table is to use the *colspan* and *rowspan* attributes of the <td> and <th> tags.

colspan

```
<table summary="This is an empty sample table." class="mytable">
  <tr>
    <td colspan="2">This cell spans 2 columns</td>
    <td>Content</td>
  </tr>
  <tr>
    <td>Content</td>
    <td>Content</td>
    <td>Content</td>
  </tr>
</table>
```

The first row has 2 cells while the second have 3. If you forgot to take the extra cell out of the first row with the stretched cell, then the table would look funny.

rowspan

```
<table class="mytable" summary="This is an empty sample table.">
  <caption align="bottom">
    Optional Caption
  </caption>
  <tr>
    <td rowspan="2">This cell spans 2 rows</td>
    <td>Content</td>
    <td>Content</td>
  </tr>
  <tr>
    <td>Content</td>
    <td>Content</td>
  </tr>
</table>
```

The first row has 3 cells white the second have 2. If you forgot to take the extra cell out of the second row, then the table would look like funny.