Study Unit 2: INTRODUCTION OF HTML AND CSS

Outline

Students will be introduced to Basic HTML Tags (Elements)
CSS properties and Attributes.

- Basic HTML
- Web Standards
- Basic CSS
- Thinking…

Learning Outcomes of Study Unit 2

Upon completion of this study unit, you should be able to learn HTML

1.1: HTML

- HTML & XHTML
- HTML Tags: title, Meta, p, h1, hr. a, img, br, comments, Em, strong, ul, ol, li
- block vs. inline
- character encoding

1.2: CSS

- Why? How?
- Link, rule
- Properties: for color, for fonts

1.3: Thinking:

- Declarative Programming

Hypertext Markup Language (HTML)

1.1: HTML

- 1993: Initial official proposed description of HTML submitted to the IETF standards group.
- 1995: HTML 2 becomes an official standard language by a publication called RFC 1866(Request for Comments is a formal Document from the IETF).
- 1996-97: HTML 3.2 standardizes various features including forms, tables, image maps, and internationalization.
- 1997: HTML 4 is proposed by W3C standard body, adding style sheets, scripting, frames, embedding objects, internationalization, and accessibility for disabilities.
- 1999: HTML 4.01 the last major version of the language is published by W3C. **A majority of pages on the Web today still use it as their started language**.
- 2001-01: XHTML, HTML based on XML

Hypertext Markup Language (HTML)

- Describes the **content** and **structure** *of information* on a Web page
  - Not the same as the presentation (appearance on screen)
- Surrounds text contents with opening and closing **tags**
- Each tag's name is called an **element**
  - Syntax: <*element*> *content* </*element*>
  - Example: This is an paragraph
- Most whitespace is insignificant in HTML (ignored or collapsed to a single space)
- We will use a Transitional, more standard version called XHTML

- HTML coding convention: the structure of HTML is a tree.
  - Indent nested elements
  - Separate siblings with blank line when it makes reading easy.

- Responsibility of HTML languages
  - HTML describes the content and structure
  - Style Sheets (CSS) describes the appearance of the document
  - Script (JavaScript) describes the behavior of the document

- index.html
- By Default Page is index.html

Comments: <!-- … -->

Comments to document your HTML file or "omment out" text

```
<!-- My web page, by Tim Student SS 12345, Spring 2048 -->
<p>SS courses are <!-- NOT --> a lot of fun!</p>
```

HTML

SS courses are a lot of fun!

output

- Comments are still useful for disabling sections a page
- Comments cannot be nested and cannot contain a –
- Many web pages are not thoroughly commented (or at all)
  - Comment is a communicative approach, to explain your designs and purposes to your colleagues, or even yourself sometime later.
  - Comment is not for browsers of end users, but for developers and designers.

Structure of a XHTML Page

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        information about the page
    </head>
    <body>
        page contents
    </body>
</html>
```

XHTML

- The header describes the page and the body contains the page's contents
- An HTML page is saved into a file ending with extension .html

Page Title: <title>

Describes the title of the Web page

```
<title>Chapter 2: HTML Basics</title>
```

HTML

- Placed within the **head** of the page
- Displayed in the Web browser's title bar and when bookmarking the page

Page Meta Data: <meta>

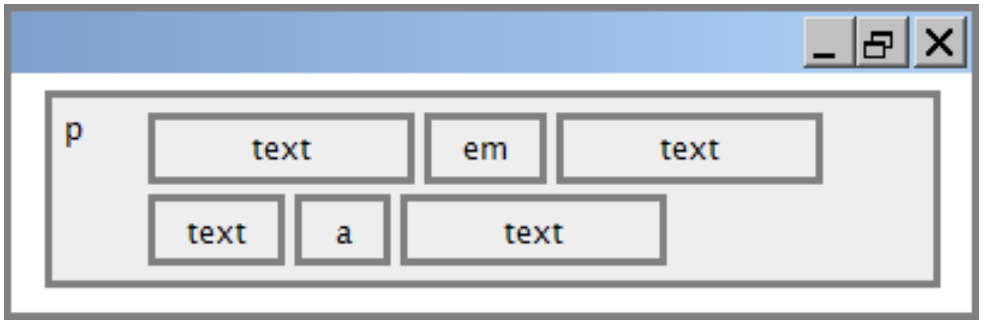
Describe meta data of the Web page

```HTML
<meta name="description" content="introduction of KIU" />
```

```HTML
<meta http-equiv="Content-Type" content="text/html;
charset=gbk" />
```

- Placed within the **head** of the page
- **Chartset** is very significant in practice, and we often use **utf-8** for language other than English

Block and Inline Elements



- **Block** elements contains an entire large region of content
  - Examples: paragraphs, lists, table cells
  - The browser places a margin of whitespace between block elements for separation
- **Inline** elements effects a small amount of content
  - Examples: bold text, code fragments, images
  - The browser allows many inline elements to appear on the same line
  - Must be nested inside a block element

Paragraph: <p>

```
<p>You're not your job. You're not how much money you have
in the bank. You're not the car you drive. You're not the
contents of your wallet. You're not your khakis. You're
the all-singing, all-dancing crap of the world.</p>
```

HTML

You're not your job. You're not how much money you have in the bank. You're not the car you drive. You're not the contents of your wallet. You're not your khakis. You're the all-singing, all-dancing crap of the world.

output

- Placed within the **body** of the page
- More paragraph examples

Line Break:

Forces a line break in the middle of a block element (*inline*)

```
<p>Teddy said it was a hat, <br /> So I put it on.</p> <p>Now Daddy's sayin', <br /> Where the heck's the toilet plunger gone?</p>
```
HTML

Teddy said it was a hat,
So I put it on.
Now Daddy's sayin',
Where the heck's the toilet plunger gone?

output

- **br** should be immediately closed with **/>**

Headings: <h1>, <h2>, …<h6>

Headings to separate major areas of the page (*block*)

```
<h1>Kampala International
University</h1>
<h2>School of Software</h2>
```

HTML

Kampala international University

School of Software

Support by Google

output

- More heading examples

Semantic HTML

- If you find the following code is shown too big in your browser, what will you do?

```
<h1> KIU University</h1>
```
HTML

KIU University

output

- Make it from **h1** to **h3**?
- Semantic HTML – **Separation of concerns**
  - Choosing tags based on the meaning of the content rather than its appearance
  - Flexible and reusable

Horizontal Rule: <hr />

Headings to separate major areas of the page (*block*)

```
<p>First paragraph</p>
<hr />
<p>Second paragraph</p>
```

HTML

First paragraph

Second paragraph

output

- Should immediately closed with />

More about HTML Tags

- Some tags can contain additional information called attributes
  - Syntax:
    ```
    <element attribute1="value1" attribut2="vaule2">
    content</element>
    ```
  - Example: `<a href="page2.html">Next page</a>`
- Some tags don't contain content; can be opened and closed in one tag
  - Syntax: `<element attribute1="v1" attribut2="v2" />`
  - Example: `<hr />`
  - Example: `<img src="bunny.jpg" alt="bunny" />`

Links: <a>

Links, or “anchors”, to other pages (*inline*)

```
<p>
  Search
  <a href="http://www.google.com/">Google</a> or
our
  <a href="lectures.html">Lecture Notes</a>
</p>
```

HTML

Search Google or our Lecture Notes.

output

- Uses the **href** attribute to specify the destination URL
  - Can be **absolute** (to another Web site) or **relative** (to another page on this site)
- Anchors are inline elements; must be placed in a block element such as p or h1

Links: <a>

- Hover a link in a browser, its destination URL will be shown on the status bar
- Be descriptiveness!

Click here to check your course schedule *output*

Please check your course schedule *output*

Course Schedule (please check yours before March 15!)" *output*

- What's principle applied here?
- Kind.
  - You are kind to your Web page readers by making the page descriptive, which in turn let them understand easier.

Images: <img>

Inserts a graphical image into the page (*inline*)

```
<img src="imges/gollum.jpg" alt="Gollum form LOTR" />
```

HTML

output

- The src attribute specifies the image URL
- XHTML also requires an alt attribute describing the image

More about Images

```
<a href="http://theonering.net/">
    <img src="images/gandalf.jpg" alt="Gandalf"
      title="You shall not pass!" />
</a>
```

HTML

output

- If placed inside an anchor, the image will become a link
- The **title** attribute specifies an optional tooltip
- `images/gandalf.jpg` vs. `/images/gandalf.jpg`

Phrase Elements: ,

em: emphasized text (usually rendered in *italic*)

strong: strongly emphasized text (usually rendered in **bold**)

```
<p>
  HTML is <em>really</em>,
  <strong>REALLY</strong> fun!
</p>
```

HTML

HTML is *really*, **REALLY** fun!

output

- As usual, the tags must be properly nested for a valid page
- em vs. i, strong vs. b
  - SoC again~!

Nesting Tags

Bad:

```
<p>
  HTML is <em>really,
  <strong>REALLY</em> lots of </strong>
fun!
</p>
```

HTML

- Tags must be correctly nested
  - a closing tag must match the most recently opened tag
- The browser may render it correctly anyway, but it is invalid XHTML

Unordered List: <ul>,

ul represents a bulleted list of items (*block*)

li represents a single item within the list (*block*)

```
<ul>
  <li>No shoes</li>
  <li>No shirt</li>
  <li>No problem!</li>
</ul>
```

HTML

- No shoes
- No shirt
- No problem!

output

More about Unordered Lists

a list can contain other lists:

```
<ul>
  <li>Simpsons:
    <ul>
      <li>Homer</li>
      <li>Marge</li>
    </ul>
  </li>
  <li>Family Guy:
    <ul>
      <li>Peter</li>
      <li>Lois</li>
    </ul>
  </li>
</ul>
```

HTML

- Simpsons:
  - Homer
  - Marge
- Family Guy:
  - Peter
  - Lois

output

Ordered List:

ol represents a numbered list of items (*block*)

```
<p>RIAA business model:</p>
<ol>
  <li>Sue customers for copying music</li>
  <li>???</li>
  <li>Profit!</li>
</ol>
```

HTML

RIAA business model:

1. Sue customers for copying music
2. ???
3. Profit!

output

Outline

- Basic HTML
- **Web Standards**
- Basic CSS
- Thinking…

Web Standards

- It is important to write proper XHTML code and follow proper syntax.
- Why use XHTML and Web standards?
  - More rigid and structured language
  - More interoperable across different web browsers
  - More likely that our pages will display correctly in the future

XHTML Versions

- *XHTML 1.0* (W3C Recommendation)
  - HTML 4.01 with XML syntax
  - XHTML 1.0 Strict, XHTML 1.0 Transitional, XHTML 1.0 Frameset
- *XHTML 1.1* (W3C Recommendation)
  - Module-based XHTML
  - Ruby characters
- *XHTML 1.2*
  - Improved Semantic Web support through RDFa
  - Draft, not widely adopted
- *XHTML 2.0*
  - Not backward compatible
  - Only in draft, not standard, and will be expired at the end of 2009
- XHTML 5
  - Part of HTML5 specification (ongoing)

XHTML 1.0 vs HTML 4.01

- All tags must be closed
- All tags must be correctly nested
- All tag attributes must be enclosed in quotation marks
- The & character can't be used on its own, and use & instead
- Tags are case-sensitive and must be all in lowercase
- Attributes cannot be minimized any more
- XHTML document must start with new XML declaration
  - `<?xml version="1.0" encoding="UTF-8"?>`
- Different DOCTYPE declaration
  - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
- The <html> tag requires an xmlns attribute

W3C XHTML Validator

```
<p>
  <a href="http://validator.w3.org/check/referer">
    <img src="http://www.w3.org/Icons/valid-
xhtml11" alt="Validate" />
  </a>
</p>
```

HTML

output

- validator.w3.org
- Check your HTML code to make sure it follows the official XHTML syntax
- More picky than the browser, which may render bad XHTML correctly

Outline

- 1.1: Basic HTML
- Web Standards
- **1.2: Basic CSS**
- Thinking…

The Bad Way to Produce Styles

```
<p>
  <font face="Arial">Welcome to Greasy Joe's.</font>
     You will <b>never</b>, <i>ever</i>, <u>EVER</u>
     beat <font size="+4" color="red">OUR</font> prices!
</p>
```

HTML

Welcome to Greasy Joe's. You will **never**, *ever*, EVER beat OUR prices!

output

- Tags such as b, i, u, and font are discouraged in strict XHTML
  - Why it is bad?

Cascading Style Sheets (CSS): <link>

```
<head>
   <style type="text/css"  media="screen">
   ...
   </style>
</head>
```

Embedded in HTML

```
<head>
   ...
  <link href="filename" type="text/css" rel="stylesheet"
media="screen" />
   ...
</head>
```

standalone CSS file

- **CSS** describes the appearance and layout
  - As opposed to HTML, which describes content of the page
  - Either on screen and on print
- Can be embedded in HTML or placed into separate .css file (preferred)

Basic CSS Rule Syntax

```
 selector {
property 1: value 1;
 …
property n: value n;
 }
```
CSS

```
 p {
font-family: sans-serif;
  color:red;
 }
```
CSS

- Comments : /* …. */
- A CSS file consists of one or more **rules**
- Each rule starts with a **selector** that specifies an HTML element(s) and then applies style **properties** to them
  - A selector of * selects all elements

CSS Properties for Colors

```css
p {
  color: red;
  background-color: yellow;
}
```

CSS

This paragraph uses style above

output

property	description
color	color of the element's text
background-color	color that will appear behind the element

Specifying Colors

```
p { color: red; }
h2 { color: rgb(128, 0, 196); }
h4 { color: #FF8800; }
```

CSS

This paragraph uses the first style above.

This h2 uses the second style above.

This h4 uses the third style above.

output

color names: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, (white), yellow

- RGB codes: red, green, and blue values from 0 (none) to 255 (full)
- Hex codes: RGB values in base-16 from 00 (0, none) to FF (255, full)

CSS Properties for Fonts

Property	Description
font-family	which font will be used
font-size	how large the letters will be drawn
font-style	used to enable/disable italic style
font-weight	used to enable/disable bold style
Complete list of font properties	

font-family

```css
p { font-family; Georgia; }
h2 { font-family: "Courier New"; }
```

CSS

This paragraph uses the Georgia font.

This h2 uses the Courier New font.

output

- All Fonts are not compatible with all Browsers
  - IE can support other fonts Windows OS supported

More about font-family

```
p {
  font-family: Garamond, "Times New Roman", serif;
 }
```
CSS

If no Garamond then uses TNR, and then uses serif.

output

- Enclose multi-word font names in quotes
- Can specify multiple fonts from highest to lowest priority
- Generic font names:
  - serif, sans-serif, monospace, cursive

font-size

```css
p {
  font-size: 20pt;
 }
```

CSS

This paragraph uses font size 20pt.

output

- **Units**: pixels (px) vs. point(pt) vs. m-size(em)
  - 16px, 16pt, 1.16em
- **Vague font size**: xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger
- **Percentage font sizes**, e.g.: 90%, 120%

font-weight, font-style

```css
p {
  font-weight: bold;
  font-style: italic
 }
```

CSS

This paragraph is bold and italic.

output

- Either of the above can be set to normal to turn them off

Outline

- Basic HTML
- Web Standards
- Basic CSS
- **1.3: Thinking…**

Thinking ...

- What's the difference?

```html
<html>
   ….
<body>
  <h1>Supper Man</h1>
  <p>
    The guy teaches you Web.
  </p>
 ….
```

HTML

```php
<?php
  $file="1.txt";
  $fp=fopen($file,"r");
  $content= fread(
    $fp,filesize($file));
  fkclose($fp);
?>
```

PHP

```css
body {
  background-color: #997788;
  font-family: SimSon;
}
h1 {
  color: blue
}
```

CSS

Declarative Programming

- Imperative vs. Declarative
  - **Declarative programming** is a programming paradigm that expresses the logic of a computation without describing its control flow.
    -- Lloyd, J.W., *Practical Advantages of Declarative Programming*
  - In contrast with imperative programming, which requires an explicitly provided algorithm.

Imperative programming

A programming language that requires programming discipline such as C/C++, Java, COBOL, FORTRAN, Perl and JavaScript. Programmers writing in such languages must develop a proper order of actions in order to solve the problem, based on a knowledge of data processing and programming.

Declarative programming

A computer language that does not require writing traditional programming logic; Users concentrate on defining the input and output rather than the program steps required in a procedural programming language such as C++ or Java.
Declarative programming examples are CSS, HTML, XML, XSLT, RegX.

Self-Review Questions (SRQ) For Study Session 2

Now that you have completed this study unit, you can assess how well you have achieved its Learning Outcomes by answering these questions. Write your answers in your Study Diary and discuss them with your Tutor at the next Study Support Meeting or Online interactive sessions. You can also check your answers at the Self-Review Answers section which is located at the end of this Module.

1. What is HTML? 2. What is the difference between html and xhtml? 3. What is CSS and advantages of CSS?	4. What is the difference between declarative and imperative programming? 5. Write a web page in Html. which contains your self- introduction, recent photos, and courses selected this semester, and favorite movies.

Self-Review Answers (SRA) for Study Unit 2

1: What is HTML?

Hypertext Markup Language, a standardized system for tagging text files to achieve font, colour, graphic, and hyperlink effects on World Wide Web pages.

2: What is the difference between html and xhtml?

XHTML is not so much different from HTML 4.01 standard. The major differences are:

XHTML elements must be properly nested.

XHTML elements must always be closed.

XHTML elements must be in lowercase.

XHTML documents must have one root element.

3: What is CSS and advantages of CSS?

Cascading **S**tyle **S**heets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.
CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs,variations in display for different devices and screen sizes as well as a variety of other effects.
CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

Advantages of CSS

CSS saves time − You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.

Pages load faster − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.

Easy maintenance − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

Superior styles to HTML − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

Multiple Device Compatibility − Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.

Global web standards − Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

4: What is the difference between declarative and imperative programming?

Imperative programming

A programming language that requires programming discipline such as C/C++, Java, COBOL, FORTRAN, Perl and JavaScript. Programmers writing in such languages must develop a proper order of actions in order to solve the problem, based on a knowledge of data processing and programming.

Declarative programming

A computer language that does not require writing traditional programming logic; Users concentrate on defining the input and output rather than the program steps required in a procedural programming language such as C++ or Java.

Declarative programming examples are CSS, HTML, XML, XSLT, RegX.

5: Write a web page in Html. which contains your self- introduction, recent photos, and courses selected this semester, and favorite movies.

For practice use following website …..

http://www.w3schools.com/

References and Additional Reading Materials

- http://en.wikipedia.org/wiki/XHTML
- http://en.wikipedia.org/wiki/Cascading_Style_Sheets
- List of all HTML tags: http://www.w3schools.com/tags/default.asp
- XHTML 1.1 Spec. http://www.w3.org/TR/xhtml11/
- Fonts of each operating systems: https://www.w3schools.com/css/default.asp