Lecture 2 – HTML

Hypertext markup language or HTML is the language web pages are encoded in. This lecture presents the core elements of HTML that you will need for this course. The browser allows you to view the HTML sources of web pages you visit. Firefox even adds syntax-sensitive coloring. We encourage you to extend your knowledge of HTML beyond what is covered here by using this feature.

On standards

The *World Wide Web Consortium* a.k.a. *W3C* is the organization responsible for defining the HTML standard. Like most standards, HTML is going through revisions. Older versions of the standard are made obsolete by newer versions. HTML allows for easy extensibility so various flavors of HTML coexist on the web. If a document uses extensions from a newer version of the standard than what the browser understands (or non-standard extensions introduced by another browser), the browser just ignores the extensions and renders the document best it can. HTML is specifically designed for *forward compatibility* (new browsers can display documents confirming to older versions of the standard) and *backward compatibility* (old browsers can display with some limitations documents using a newer version of the standard).

In practice the evolution of HTML (and other web standards) is not driven solely by W3C, the dominant browser implementations act as *de facto standards*. Writers of web pages may use non-standard extensions implemented by popular browsers and they typically avoid using standard features that are not widely supported or are supported inconsistently by different browsers. Popular non-standard extensions often end up being incorporated in newer versions of the standard. While standards-compliance is still an issue, today's browsers are much better at following standards than those from the early days of the web. Browsers are written so that they can handle incorrect documents also: if the web page author makes a mistake, the browser tries to guess what he actually meant and often gets it right. In this class we will focus on writing standard-compliant web pages. You should test your web pages with W3C's free HTML validator service available at `http://validator.w3.org/`. On the next page there is a standards-compliant version of the simple web page presented in the last lecture (but both of them render identically).

Like all standards, HTML has been shaped by the often diverging priorities of various stakeholders. The most long-lived conflict with a major impact on HTML is that between standard bodies that want to ensure a certain generality for web documents and web page authors who want to strictly control the appearance of their pages. Browsers with varying window sizes and resolutions (possibly some on mobile devices) should be able to display the page. Users with different preferences in terms of font sizes should be able to see the same document. Some browsers cannot display images and are entirely text-based. People with disabilities may rely on very special browsers (e.g. the browser may read aloud the page for blind people). It is also desirable that various programs be able to

understand the structure of the document easily. But this need for generality sometimes makes it hard for web authors to write pages with visual appeal to the typical user they are targeting. As a result of this tension there is a strong drive towards separating *content* from *presentation*. This separation is far from total in today's HTML, but it is much cleaner than in early versions of HTML. A major step in this direction has been the introduction of cascading style sheets (CSS) which hold the presentation information allowing the HTML document to keep mostly content.

W3C is moving away from HTML to the more general XML-based *XHTML* (which stands for "extensible hypertext markup language"). The HTML standard defines explicitly the supported tags and attributes and their meanings. *XML* which is an acronym for *extensible markup language* does not specify in the standard what tags and attributes are allowed. It is up to a document called XML Schema or DTD (which stands for document type definition) to define these things. Thus XML is used as a general-purpose format for structured data (not just web pages, any type of structured data). Thus unlike HTML XML focuses only on the data and not on the presentation. The last HTML standard is 4.01 and it came out in 1999. The current standard is XHTML 1.0 and its latest revision came out in 2002. The language is not defined in a single standard, there is a main standard defining the core of the language and separate standards exist for various parts of the language.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head><title>Bucky Badger's web page</title></head>
<body>
<h1>Welcome to Bucky's web page</h1>
<!-- Users don't see this comment. -->
<img src="bucky.gif" alt="A picture of Bucky" />
<p>I am Bucky, the mascot for University of Wisconsin athletics.
Please visit
<a  href="http://www.uwbadgers.com/football/index.html"> the web
page of our football team</a> and
<a  href="http://www.uwbadgers.com/basketball/index.html"> the
web page of our basketball team</a>.</p>
</body>
</html>
```

Basic HTML

HTML is a text-based protocol that uses human-readable `tags` to convey document structure and formatting information. Normally the text in the HTML document is displayed by the browser with simple reformatting. To be able to accommodate various widths for the browser window, HTML mandates that the browser rearrange the text to fit the window: line breaks are introduced where the text reaches the right edge of the window (i.e. the text wraps), line breaks and tabs in the HTML document are treated as simple spaces. Not all text in the HTML document is displayed. Text enclosed between `<!--` and `-->` is represents comments that are not rendered by the browser.

The fundamental building block of HTML is the markup *tag*. Most tags come in pairs: and opening and a closing tag and they express something about the text between them. For example `<h1>Mike's ice cream page</h1>` indicates that the text between the h1 tags should be rendered as a level 1 heading. Tags can have *attributes* and some tags do not have a corresponding closing tag: `<tagname attr1="val1" attr2="val2">`. The HTML standard specifies which tags need a closing tag and which don't. Because the XML does not specify tag names, the possibility of confusion arises between omitting the closing tag by error and using a tag that does not require a closing tag. For such tags the XML syntax is `<tagname />` and this rule also applies to XHTML documents. Another difference is that for some attributes HTML does not require a value. Thus `<option selected>` is valid in HTML, but in XHTML we have to use `<option selected="selected">` because the `selected` attribute of the `option` tag must have a value. Open-close tags can be nested but they cannot overlap. Thus `<b><i>one</i>two</b>` is valid, but `<b><i>one</b>two</i>` is not.

Certain characters such as "<", ">" and "&" cannot appear inside the text of the document because the browser may interpret them as tags. Special alphabetic escape codes (`&lt;`, `&gt;` and `&amp;`) or numeric codes (`&#60;`, `&#62;` and `&#38;`) should be used instead. Such codes are also available for other commonly used non-standard characters, letters from other alphabets, mathematical symbols, etc. For example `&copy;` stands for "©" `&reg;` for "®" and ` ` for non-breaking space. You can find the full list of such codes at http://www.w3.org/TR/html401/sgml/entities.html.

An XHTML document must start with a DOCTYPE declaration which identifies the version of the standard the document follows. This line allows the browser to interpret the document correctly if different versions of the standard prescribe different interpretations for a given element. The web page is enclosed by `<html></html>` tags. Elements that do not need to be rendered inside the browser window are further enclosed by `<head></head>` tags and those get rendered inside the window are enclosed by `<body></body>` tags.

The author of a web page needs to use tags to control the flow of the text. `<br />` introduces a line break. `<p></p>` defines a paragraph. Browsers typically leave an empty line between paragraphs. The text between `<pre></pre>` tags is interpreted as pre-formatted text. Such text is rendered with a monospaced font and the browser respects new lines and tabs that are part of such text.

There are numerous tags that affect the appearance of the text they enclose. They can be categorized as content-based or physical. Content-based style tags tell the browser what the text is, whereas physical styles tell it how to render the text. The first category includes `<h1></h1>` to `<h6></h6>` for various levels of headings, `<em></em>` for emphasis `<code></code>` for program code and `<address></address>` for addresses. The second category include `<b></b>` for bold `<i></i>` for italic `<tt></tt>` for using a monospace font, `<sub></sub>` for subscript `<sup></sup>` for superscript and the

obsolete tag with its many attributes that you should never use because style sheets are a better way of formatting text.

A useful family of tags is that defining lists. defines bulleted (unordered) lists and defines numbered (ordered) lists. For ordered lists the attribute type which can take the values A, a, 1, I, or i controls whether the browser uses letters, numbers or roman numerals (upper or lower case). The attribute start controls the label of the first item of an ordered list. List items are defined by . Lists can be nested and the browser uses indentation and different bullet types to convey the nesting.

More on URLs and links

The <a> tag is used to define hyperlinks. The enclosed text becomes the link the user can click on and the href attribute specifies the destination URL the link points to. The URL in a link needs not specify all four parts of the URL discussed in the last lecture. The missing parts are taken from the URL of the document that includes the link, called the *base URL*. For example if a document located at http://foo.com/dir/page.html has a link that specifies href="otherpage.html" the actual URL the link will point to will be http://foo.com/dir/otherpage.html. Similarly subdir/page2.html points to http://foo.com/dir/subdir/page2.html, and /otherdir/page3.html to http://foo.com/otherdir/page3.html. These types of partial URLs are called *relative URLs* as opposed to the full URLs called *absolute URLs*. Relative URLs are not just a convenience feature for lazy web page authors; they make it possible to move large collections of documents pointing to each other from one server to another (or to a different directory within the same server) without rewriting them.

There are some "incomplete" URLs that are not relative URLs. For example the file name can be missing. When the web server receives such a URL it looks for a default file (for some web servers it is "default.htm", for others "index.html") and if it finds it, it sends it to the browser. If the web server does not find such a file it builds a web page with the list of files in the directory and sends it to the browser. Also, URLs may use the fake protocol "file://" which instructs the browser to read the document from the local file system of the computer the browser is running on. Naturally for this "protocol" it does not make sense to specify the server name since the document is local. URLs using the "mailto:" protocol are even more different from regular URLs. This protocol is followed by an email address and the browser reacts to clicks on links with this type of URL by opening an email client window for sending a message to that address.

Links can point not just to a document, but also to a specific element within a document. Thus href="page.html#conclusions" points to an element of the page named "conclusions". When the user clicks on this link, the browser will scroll the document to ensure that the target page element is visible. There are two methods for giving a page element a name so that URLs can point to it. One can enclose the element within an <a> tag and use the name attribute (usually without the href attribute). Alternatively almost all tags support the id attribute and its value can also be used. For example the target for the link above can be <h2 id="conclusions">Final thoughts</h2>. By

setting the `target` attribute of the `<a></a>` tag to `_blank` the web page writer can ask the browser to open the linked document in a new window or new tab when the user clicks on the link (instead of replacing the current document in the current window).

Forms

HTML forms are the traditional method for users sending information to a web server. The user fills out some fields, submits them (usually by clicking on a button called submit), http carries the user data to the server, a server-side program processes it and produces a document that is sent back to the browser. The `<form></form>` tag defines forms in the document. It is forbidden to nest multiple `<form></form>` tags. This tag has two important attributes: `action` which has the URL of the server side program that processes that data the user submits and `method` which has to have the value `"get"` or `"post"` and defines how user data is carried to the server (more on this when we discuss http). Within the `<form></form>` tags, the page typically has other tags defining various *controls* that collect input from the user, explanatory text and some tags controlling formatting and layout in the browser. The server receives user data as a sequence of `FieldName=FieldValue` pairs corresponding to controls inside the form.

The `<input />` tag is used for defining most types of controls used in HTML. It has three important attributes: `type`, `name` and `value`. If the `type` attribute is `"text"` the tag defines a text field where the user can enter input. If `type` is `"submit"` the tag generates a submit button and if it is `"reset"` the button resets all controls within the form to their initial values erasing all data entered by the user. If `type` is `"checkbox"`, the tag generates a checkbox and if it is `"radio"` it generates a radio button. Note that while checkboxes can be set and reset independently, the browser ensures that within a group of radio buttons, a single one is set at any given time (by analogy to buttons on old radios using mechanical push buttons to select the station playing – when you pushed the button for another station, the button of the station currently playing popped out automatically). When the `type` attribute has the value `"hidden"`, the browser doesn't render anything, but the information associated with the control is sent to the server. Hidden controls can be used to submit preferences that the user needs not select explicitly. Later in this course we will see that hidden controls have other surprising uses. The `name` attribute is used to define the "field name" associated with the data generated by the control when it is submitted to the server. All controls except submit and reset buttons must have this attribute. It is not required that different controls within a form have different names. For example radio buttons that work together as a group must have the same name. The `value` attribute acts differently for different controls. For checkboxes and radio buttons it represents the value submitted to the server, for buttons it specifies the text displayed on them, and for text controls it is the initial value of the text field.

The `<textarea></textarea>` tag is used for multi-line text areas for user input. The text between the tags is the initial in the text area. The `cols` and `rows` attributes specify the size of the size of the text area. The `<select></select>` tag generates a drop-down list from which the user can select one option. If the `multiple` attribute is present, the user

can select more than one option. The `<option></option>` tag is used to indicate the options from the list.

Tables

HTML tables are also used for displaying tables, but their most prevalent use is for controlling the placement of various elements on the page (by `http://www.google.com/` and `http://www.cs.wisc.edu/` for example)[1]. An HTML table consists of rows, and rows consist of cells with cells on the same row/column being aligned. Cells can contain anything not just text: images, forms, other tables, etc. Since the "grid" defined by columns and rows can be hidden from the users, tables can be used to control alignment between various regions of the web page.

The `border` attribute of the `<table></table>` tag defines the width in pixels for the lines to draw the table. If its value is 0, no lines are drawn. The `width` attribute controls the width of the entire table. If the value is a number it is interpreted as the number of pixels, if it is a number followed by "%", it is interpreted as a percentage of the page width. If this attribute is not specified, the table is made as narrow as possible to fit the data inside. Text is wrapped to keep the table growing wider than the window.

The `<tr></tr>` tag is used to define rows within the table. It has the `nowrap` attribute that instructs the browser not to wrap text within the cells of the row and it also has attributes for controlling the vertical and horizontal alignment of text within the cells. The `<td></td>` tag is used to define cells of the table. It has a width attribute which defines cell width as a fraction of the table width when given as a percentage. A cell can span multiple columns and/or multiple rows if the `colspan` or `rowspan` attributes are used.

Note that it is possible to write tables that give contradictory information to the browser (e.g. the widths of the columns expressed in pixels are larger than the width of the table expressed in pixels). When browsers receive such contradictory instructions they override some of them and typically produce a good rendering of the table.

[1] Frames and iframes can be used for controlling placement, but you should not use them because they behave inconsistently on different browsers and they break the one-to-one relationship between documents displayed to the user and URLs.