HTML5

Objectives

- Learn how markup languages like HTML process documents
- Understand HTML syntax and structure
- Learn basic tags and how to use them
- Know the differences between HTML5 and earlier versions

HTML History and Basic Syntax

HTML History

HTML defined using **SGML**

- Used for general layout of documents that could be displayed by a wide variety of computers

HTML versions:

- HTML 4.0 (1997)
- HTML 4.01 (1999)
- XHTML 1.0 (2000): HTML 4.01 redefined using XML
- XHTML 1.1 (2001): Modularized 1.0, no frames
- XHTML 2.0: 2009, dropped from development
- **HTML5**: newest version

Basic HTML Syntax

Elements defined by ***tags***:

- Opening tag: <name>
- Closing tag: </name>

Container := opening tag + closing tag

Container encloses ***content***

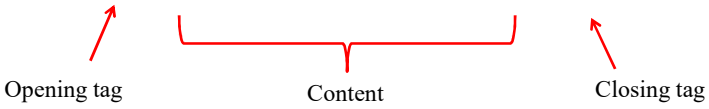
Element := container + content

Void element := tag without content

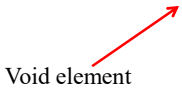
- Written as: <name />

HTML Syntax Examples

```
<p>This is my web page</p>
```




```
<br />
```



XHTML vs HTML5:
XHTML required
HTML5 recommends <br> but support either

We will utilize <br> in our labs and assignments

Tag Attributes

Tags may have ***attributes*** that provide additional information to the browser on how the tag should be interpreted

Attributes appear between the tag name and the right bracket of the opening tag

Attributes are specified in **keyword form**, delimited by double quotes

```
attrName1="value1" attrName2="value2"
```

Example:

```
<img src="picture.jpg" alt="Picture">
```

(src and alt are attributes, "picture.jpg" and "Picture" are the attribute values)

Coding Style

Coding Style

Whitespace := line breaks, spaces, tabs that browsers do not process

Comments := text that describes the code (HTML), but is not processed by the browser

- Comment form: <!-- … -->
- Example: <!-- This is the first section -->

Make use of both whitespace and comments to make your code easily readable and understandable!

- Consistently indent (tab or 2 or 4 spaces) each nested element
- Separate sections of code with blank lines
- Insert comments with good descriptions around sections of code
- Insert comments whenever a line of code may need clarification

Coding Style Examples

Bad style example:

```
<!DOCTYPE html>
<html lang="en">
<head><title>This is the title
</title></head><body>
<p>This is my webpage</p><img src="pic.jpg" alt="picture" /><a
href="www.google.com"> Link
<img src="link.jpg" alt="Link to google" /></a>
<h1>Section1</h1><p>This is the first section</p>
<br /></body></html>
```

Coding Style Examples

Good style example:

```
<!DOCTYPE html>

<!--
      This is my personal webpage
      Modified: 09/06/2012
-->
<html lang="en">

  <head>
    <title>This is the title</title>
  </head>

  <body>
    <-- Initial information -->
    <p>This is my webpage</p>
    <img src="pic.jpg" alt="picture" />

    <-- Link with an image to favorite website -->
    <a href="www.google.com">
      Link
      <img src="link.jpg" alt="Link to google" />
    </a>

    <-- Beginning of the first section -->
    <h1>Section1</h1>
    <p>This is the first section</p><br />

  </body>
</html>
```

HTML Document Structure and Basic Tags

HTML5 Document Structure

Every HTML5 document should begin with:

<!DOCTYPE html>

<html>, <head>, <title>, <body> are required elements

<html> is the root of the whole document

- Should include *lang* attribute: <html lang="en">
- Inside of <html> should be the document head and the body: <head>, <body> tags

Head of the document (inside **<head>** tag):

- Include **<title>** tag (usually displayed in title bar)
- Include **<meta charset="utf-8" />** to provide character set information (UTF=Unicode Transformation Format - 8 bit)
- Can also contain scripts and CSS style sheets

HTML Layout Formatting

Text that will be displayed in the browser should normally be placed in ***paragraph elements***: ** ** tag

- breaks the current line and inserts a blank line, where the beginning of the content of the paragraph is placed
- Should be closed by tag

Paragraph example:

```
…
<body>
  <p>
    Welcome to my webpage!
  </p>
</body
…
```

HTML Layout Formatting

Line breaks

- The effect of the **
** tag is the same as that of <p>, except for the blank line (in HTML, it could be just <br>)
- No closing tag!

Example of paragraphs and line breaks

```
On the plains of hesitation <p> bleach the
bones of countless millions </p> <br />
who, at the dawn of victory <br /> sat down
to wait, and waiting, died.
```

Typical display of this text:

```
On the plains of hesitation

bleach the bones of countless millions
who, at the dawn of victory
sat down to wait, and waiting, died.
```

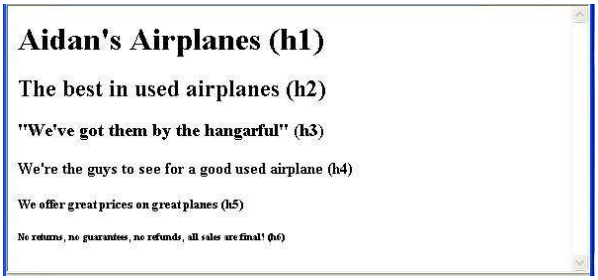
Preserving whitespace

- The text content of a **<pre>** element is displayed as it is entered

HTML Layout Formatting

Heading tags: <h1>, <h2>, …, <h6>

- 6 sizes
- 1, 2, and 3 use font sizes that are larger than the default font size
- 4 uses the default size
- 5 and 6 use smaller font sizes



HTML Formatting

Deprecated Tags

- Tags that are still supported, but should not be used because they are not guaranteed to be supported in the future
- Examples: <i> (for italics), <b> (for bold)
- <i>,<b> were used to style the text, but now ***style sheets*** are used instead

Content-based style tags

- Tags that define the type of text that is to follow
- Browsers may render such text in a special way
- Examples:
  **** (emphasize, usually italics)
  **** (usually bold)
  <code> (monospaced font; used for program listings)

HTML Formatting

Supscripts and superscripts: **<sub>, <sup>**

- Example: `x<sub>2</sub><sup>3</sup>`
  Displays: x_2^3

<blockquote> tag

- Used for separating pieces of text from the normal flow of text
- Browsers usually indent, sometimes italicize
- A *block element*: should not be nested inside of *inline elements*

Formatting done by content-based style tags can be done by using style sheets, but tags are not yet deprecated

Character Entities

Character entities are used to insert special symbols that cannot be normally typed (“¼”) or have special meaning in HTML (“<“)

Char.	Entity	Meaning
&	&amp;	Ampersand
<	&lt;	Less than
>	&gt;	Greater than
"	&quot;	Double quote
'	&apos;	Single quote
¼	&frac14;	One quarter
½	&frac12;	One half
¾	&frac34;	Three quarters
°	&deg;	Degree
(space)		Non-breaking space

Horizontal Rules

<hr /> draws a line across the display, after a line break

Same thing can be done with style sheets

Meta Element Usage

Meta elements can be used to specify information for search engines

Provide additional information about a document

- e.g.: keywords, description
- Use `name` and content `attribute` pair

Examples:

```
<meta name="keywords" content="java, c, programming
language, computers">
<meta name="description" content="This is the
personal website of...">
```

Examples

Let's see some examples:

greet.html [link]
blockquote.html [link]
headings.html [link]

greet.html link:
http://www.epogue.info/cpsc-24700/Presentations/examples/w8code2/greet.html

blockquote.html link:
http://www.epogue.info/cpsc-24700/Presentations/examples/w8code2/blockquote.html

headings.html link:
http://www.epogue.info/cpsc-24700/Presentations/examples/w8code2/headings.html

Using Images

Image Formats

GIF (Graphic Interchange Format)

- 8-bit color (256 different colors)

JPEG (Joint Photographic Experts Group)

- 24-bit color (16 million different colors)

Both use compression, but JPEG compression is better

Portable Network Graphics (PNG)

- Newer
- Files are bigger than jpeg – no lost data!

Using Images

Images are inserted into a document with the **<img/>** tag along with the `src` attribute

The `alt` attribute is used for

- Non-graphical browsers
- Browsers with images turned off

Example:

```
<img src = "comets.jpg"
       alt = "Picture of comets" />
```

The `<img>` tag has 30 different attributes, including `width` and `height` (in pixels)

(width and height should always be specified to speed up page loading!)

Using Images

Make sure to specify the *relative path* with the filename inside of the *src* attribute:

src="pictures/dog.jpg"

Relative path: assumes that the current path is the document root (e.g. public_html or www directories) and looks in the specified path for the given file

e.g. **/public_html/pictures/dog.jpg**

Make sure you have permission to use the picture in your own website!

Examples

image.html [link]

image.html link:
http://www.epogue.info/cpsc-24700/Presentations/examples/w8code2/image.html

Hypertext

Hypertext Links

Hypertext is the essence of the Web!

Links are specified with the `href` (hypertext reference) attribute of the **`<a>`** (anchor) tag

Content of `<a>` is the visual link in the document

Hypertext Links

Style note: **a link should blend in with the surrounding text**, so reading it without taking the link should not be made less pleasant

Links can have images:

```
<a href = "c210data.html">
  <img src = "smallplane.jpg"
   alt = "Small picture of an airplane" />
    Info on C210 </a>
```

Hypertext Links within a Page

If the *target* is not at the beginning of the document, the target spot must be marked

Target labels can be defined in many different tags with the ***id*** attribute, as in

```
<h1 id = "baskets"> Baskets </h1>
```

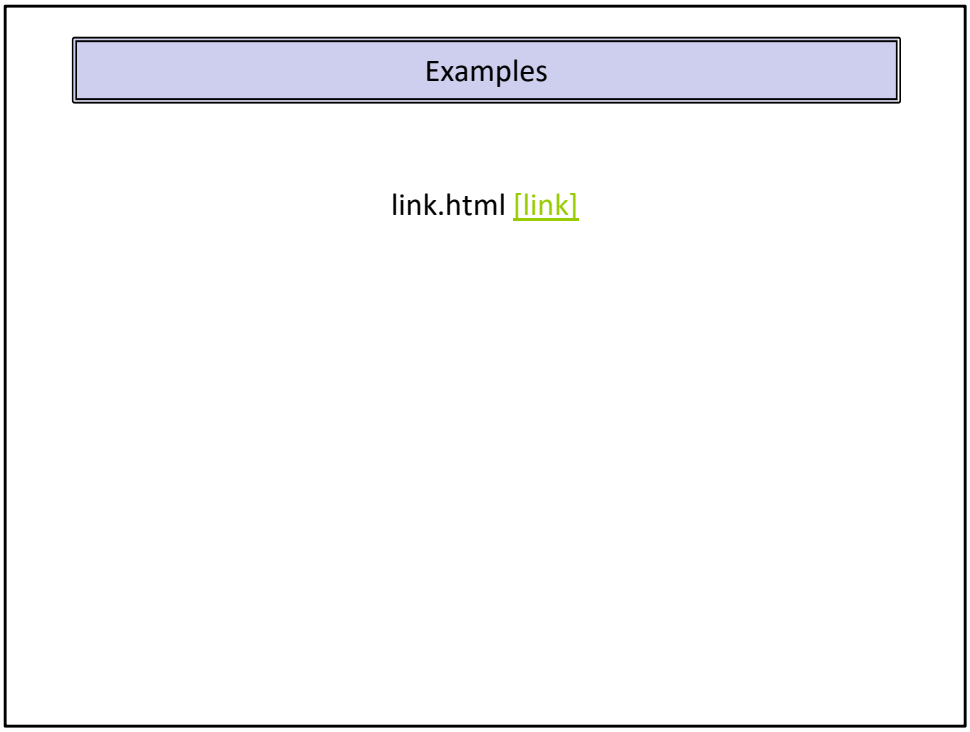
The link to an *id* must be preceded by a pound sign (#)

If the id is in the same document, this target could be

```
<a href = "#baskets"> What about baskets? </a>
```

If the target is in a different document, the document reference must be included

```
<a href = "myAd.html#baskets”> Baskets </a>
```



Link for link,html:
http://www.epogue.info/cpsc-24700/Presentations/examples/w8code2/link.html

Lists and Tables

Lists

Three types of lists:

- Unordered (starts with **<ul>** tag)
- Ordered (starts with **<ol>** tag)
- Definition list (starts with **<dl>** tag)

For unordered and ordered lists:

- **<li>** tag defines each list element
- Can nest lists, but must start new list with **<li>** tag

For definition lists:

- **<dt>** specifies the term
- **<dd>** specifies the definition

Examples

ordered.html
unordered.html
definition.html
nested_lists.html

Tables

A **table** is a matrix of cells, each possibly having content

- The cells can include almost any element
- Some cells have row or column labels and some have data

A table is specified as the content of a `<table>` tag

- Each row of a table is specified as the content of a `<tr>` tag
- The row headings are specified as the content of a `<th>` tag
- The contents of a data cell is specified as the content of a `<td>` tag

Tables - Borders and Titles

A `border` attribute in the `<table>` tag specifies a border between the cells

- If `border` is set to "`border`", the browser's default width border is used
- The `border` attribute can be set to a number, which will be the border width
- Without the `border` attribute, the table will have no lines. (unless specified by a style sheet)
- In HTML5, table borders should be defined using style sheets

Tables are given titles with the **`<caption>`** tag, which can immediately follow `<table>`

Tables - colspan and rowspan

A table can have two levels of column labels

- The `colspan` attribute must be set in the `<th>` tag to specify that the label must span some number of columns

The `rowspan` attribute can be set to specify that the label must span some number of rows

Tables - Sections

Header, body, and footer, which are the elements:

- `thead` (column labels)
- `tbody` (row labels + data)
- `tfoot` (usually used for totals)

If a document has multiple `tbody` elements, they are separated by thicker horizontal lines

Table Usage

In the past, tables were used to align elements in rows and columns - general layout

This is no longer done and is frowned upon

Use ***Cascading Style Sheets*** to place elements in rows and columns - general layout

Use tables only when the information is naturally tabular

Examples

table.html

cell_span.html

Forms

Forms

A **form** is the usual way information is gotten from a browser to a server

HTML has tags to create a collection of objects that implement this information gathering

- The objects are called ***widgets*** or ***controls*** or ***components*** (e.g., radio buttons and checkboxes)
- All of the controls of a form are defined in the content of a `<form>` tag

Form Processing

When the ***Submit*** button of a form is clicked, the form's values are sent to the server for processing

The only required attribute of `<form>` is **`action`**, which specifies the URL of the application that is to be called when the *Submit* button is clicked

```
action = "http://www.cs.ucp.edu/cgi-
bin/survey.pl"
```

If the form has no action, the value of `action` is the empty string

The **`method`** attribute of `<form>` specifies one of the two possible techniques of transferring the form data to the server, `get` and `post`

- **`get`** appends form data as a query string at the end of the URL
- **`post`** appends form data to the end of the HTTP request message

Form Widgets

Many are created with the **<input>** tag

The `type` attribute of `<input>` specifies the kind of widget being created:

- text
- password
- checkbox
- radio
- reset
- submit

Other widget tags

- **<select>**
- **<textarea>**

Text box

A ***Textbox*** creates a horizontal box for text input

Default size is 20; it can be changed with the `size` attribute

If more characters are entered than will fit, the box is scrolled (shifted) left

If you don't want to allow the user to type more characters than will fit, set `maxlength`, which causes excess input to be ignored

```
<input type = "text" name = "Phone"
size = "12" />
```

Labels

Widgets can be placed in label elements

```
<label> Phone: <input type = "text" name =
"phone"/>
</label>
```

Label tag makes sure that the label text is “attached” to the given widget

Password Widget

Just like text except asterisks are displayed, rather than the input characters

Checkbox

Checkboxes are used to collect multiple choice input

Every checkbox requires a `value` attribute, which is the widget's value in the form data when the checkbox is 'checked'

- A checkbox that is not 'checked' contributes no value to the form data
- By default, no checkbox is initially 'checked'

To initialize a checkbox to 'checked', the `checked` attribute must be set to `"checked"`

Radio Buttons

Radio buttons are ***collections of checkboxes in which only one button can be 'checked' at a time***

Every button in a radio button group MUST have the same name

If no button in a radio button group is 'pressed', the browser often 'presses' the first one

Selection Box

Uses the `<select>` tag

(The name attribute of `<select>` is required)

Each item of a menu is specified with an `<option>` tag, whose pure text content (no tags) is the value of the item

An `<option>` tag can include the selected attribute, which when assigned "selected“ specifies that the item is preselected

Selection box (continued)

There are **two kinds of menus**:

- those that behave like checkboxes and
- those that behave like radio buttons (the default)

Menus that behave like checkboxes are specified by including the `multiple` attribute, which must be set to "multiple"

The `size` attribute of `<select>` can be included to specify the number of menu items to be displayed (the default is 1)

(If `size` is set to > 1 or if multiple is specified, the menu is displayed as a pop-up menu)

Text Areas

Created with **<textarea>**

Usually include the `rows` and `cols` attributes to specify the size of the text area

Default text can be included as the content of `<textarea>`

Scrolling is implicit if the area is overfilled

Reset and Submit Buttons

Both are created with <input>

```
<input type = "reset"
       value = "Reset Form" />
<input type = "submit"
       value = "Submit Form" />
```

Submit has two actions:

- Encode the data of the form
- Request that the server execute the server-resident program specified as the value of the action attribute of <form>

A Submit button is required in every form

Examples

checkbox.html
menu.html
popcorn.html
radio.html
textarea.html

HTML5 Specifics

HTML5

HTML5 is still new

- Some browsers may not support it
- Some browsers support some features
- Code can be included to detect HTML5 features and produce a message to the user

Audio

Prior to HTML5: needed a plug-in for audio

Audio files use encoding algorithms (***audio codecs***, e.g. MP3, Vorbis)

Coded audio data is stored in **containers** (Ogg, MP3, Wav)

- Vorbis is stored in Ogg container
- MP3 in MP3 container
- Wav in Wav container

Inserting Audio in HTML5

Use ***audio*** element:

```
<audio attributes>
  <source src="filename1">
  …
  <source src="filename2">
  Your browser does not support the audio element
</audio>
```

Browser chooses the first audio file it can play and skips the content; if none, it displays the content

Different browsers have different audio capabilities

The `controls` attribute, which is set to "controls", creates a start/stop button, a clock, a progress slider, total time of the file, and a volume slider

Video

Prior to HTML5: needed a plug-in

Video codecs are stored in containers (just like audio)

Video codecs:

- H.264 (MPEG-4 AVC) - stored in MPEG-4 container
- Theora - any container
- VP8 - any container

Different browsers support different codecs

Inserting Video in HTML5

Use **<video>** tag

```
<video attributes>
  <source src="filename1">
  <source src="filename2">
  Your browser doesn't support video element
</video>
```

The `width` and `height` attributes set the screen size

The `autoplay` attribute, set to "autoplay", specifies that the video should play as soon as it is ready

The `preload` attribute, set to "preload", specifies that the video should be loaded as soon as the document is loaded

The `controls` attribute, set to "controls", is like that of the audio element

HTML5 Organizational Elements

Header elements

<hgroup> - container for header information

```
<hgroup>
      <header>
        <h1> The Podunk Press </h1>
        <h2> "All the news we can fit" </h2>
      </header>
      -- table of contents –
</hgroup>
```

HTML5 Organizational Elements

Footer elements

<footer> - container for footer information

```
<footer>
  &copy; The Podunk Press, 2012
  <br />
  Editor in Chief: Squeak Martin
</footer>
```

HTML5 Organizational Elements

`<section>` - container for sections

`<article>` - container for self contained parts of a document (from another source)

`<aside>` - container for tangential info

`<nav>` - navigational sections (list of links)

Examples

audio.html
organized.html
tester.html
testvideo.html

Syntactic Differences between HTML & XHTML

- Case sensitivity
- Closing tags
- Quoted attribute values
- Explicit attribute values
- `id` and `name` attributes
- Element nesting

HTML5 Validation

There are tools that exist to validate whether your HTML code adheres to the HTML5 standard

Here is one by W3Schools:
http://validator.w3.org/#validate_by_upload+with_options

Make sure to select “HTML5 (experimental)” as your document type

NOTE: All of your HTML5 code should pass validation!

Summary

- HTML5 is the newest version of a markup language used for specifying the layout of documents
- HTML uses tags to markup the content of the document
- Tags may have attributes that modify how the tag is interpreted by the browser
- It is important to maintain good programming style - comment and use whitespace
- Required elements of HTML are <html>, <head>, <title>, and <body>
- Some common formatting tags are: <p>, <br>, <h1>, <h2>, <hr>
- HTML tags can be used to insert: images, hypertext, lists, tables, and forms