CHAPTER 2: HTML

Note

- All examples for this chapter are located at

http://swe.umbc.edu/~zzaidi1/is448/chap2-examples/

- You can look at the HTML code of the examples by doing a right click on the browser window displaying the page, and selecting ‘View Source’ option.

Pop Quiz: Terminology

```
<body>
    <p> This is a pop quiz
    <img src="smiley.jpg" alt="smiley image" />
    <a href="click.html"> Click for results </a>
    <br />
    </p>
</body>
```

In the code above, identify the

- Tag names:
- Attribute names:
- Attribute values:

Lists

- Three types
  - Unordered lists <ul>
  - Ordered lists <ol>
  - Definition lists <dl>

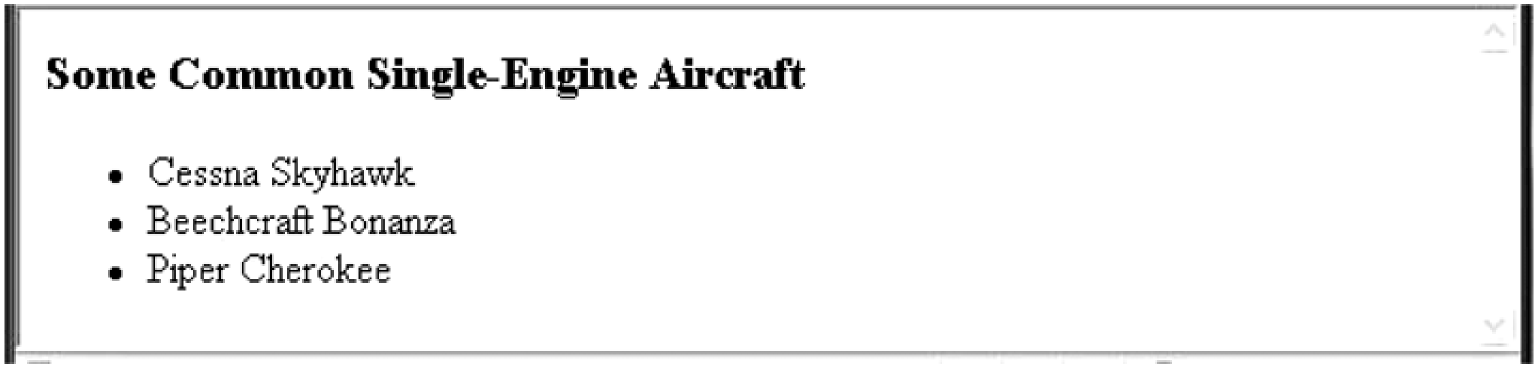
Unordered List: <ul>, <li>

- ul: represents a bulleted list
- li: represents a single item within list
- ul, li are block tags
- Example: see all_lists.html

```
<ul>
    <li> Cessna Skyhawk </li>
    <li> Beechcraft Bonanza </li>
    <li> Piper Cherokee </li>
</ul>
```

HTML Code

Browser view



Some Common Single-Engine Aircraft

- Cessna Skyhawk
- Beechcraft Bonanza
- Piper Cherokee

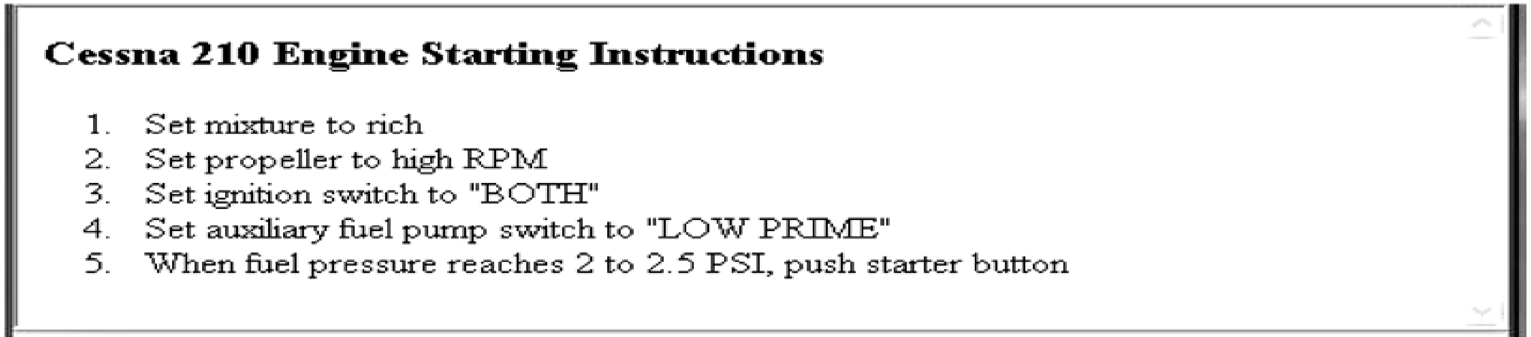
Ordered List

- ol: represents a numbered list
- li: represents a single item within list
- ol and li are block tags
- Example: see all_lists.html

HTML Code

```
<ol>
    <li> Set mixture to rich </li>
    <li> Set propeller to high RPM </li>
    <li> Set ignition switch to "BOTH" </li>
    <li> Set auxiliary fuel pump switch to "LOW PRIME" </li>
    <li> When fuel pressure reaches 2 to 2.5 PSI, push starter button
    </li>
</ol>
```

Browser view



Cessna 210 Engine Starting Instructions

1. Set mixture to rich
2. Set propeller to high RPM
3. Set ignition switch to "BOTH"
4. Set auxiliary fuel pump switch to "LOW PRIME"
5. When fuel pressure reaches 2 to 2.5 PSI, push starter button

Nested Ordered List

- <ol> tag cannot immediately follow another <ol> tag, Nested list must be within <li> tag

HTML Code

```
<ol>
   <li> General Aviation (piston-driven engines)
      <ol>
          <li> Single-Engine Aircraft </li>
          <li> Twin-Engine Aircraft </li>
       </ol>
   </li>
  <li> Commericial Aviation
     <ol>
          <li> Dual-Engine Aircraft </li>
          <li> Tri-Engine Aircraft </li>
     </ol>
   </li>
</ol>
```

Example:
See nestedlist.html

Nested lists

Definition Lists

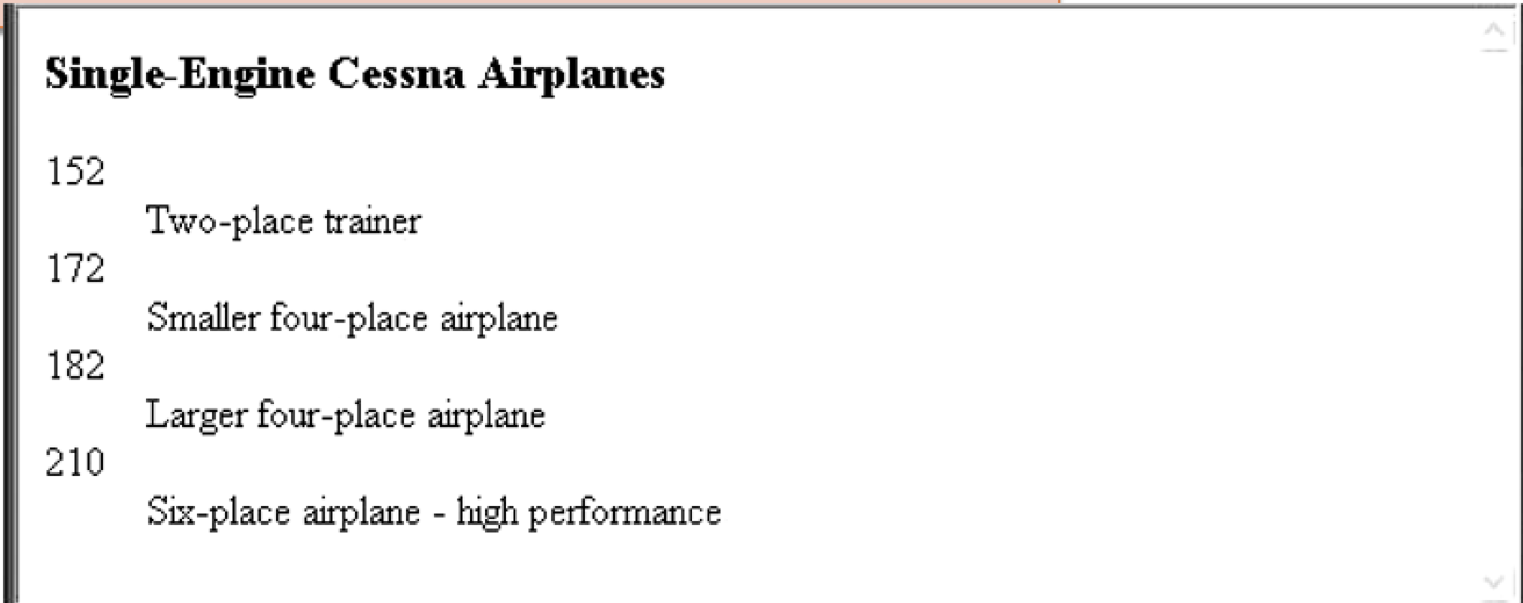
- Used to specify lists of terms and their definitions (like a glossary)
  - <dl> tag: definition list
  - <dt> tag: each term to be defined in list
  - <dd> tag: definition itself
- dl, dd are block tags
- Example: see all_lists.html

Definition Lists

```
<dl>
    <dt> 152 </dt>
        <dd> Two-place trainer </dd>
    <dt> 172 </dt>
        <dd> Smaller four-place airplane </dd>
    <dt> 182 </dt>
        <dd> Larger four-place airplane </dd>
    <dt> 210 </dt>
        <dd> Six-place airplane - high performance </dd>
</dl>
```

HTML Code

Browser view



Single-Engine Cessna Airplanes

```
152
        Two-place trainer
172
        Smaller four-place airplane
182
        Larger four-place airplane
210
        Six-place airplane - high performance
```

Tables

- Matrix of rows and columns
- Intersection of a row and a column: cell
- Typically:

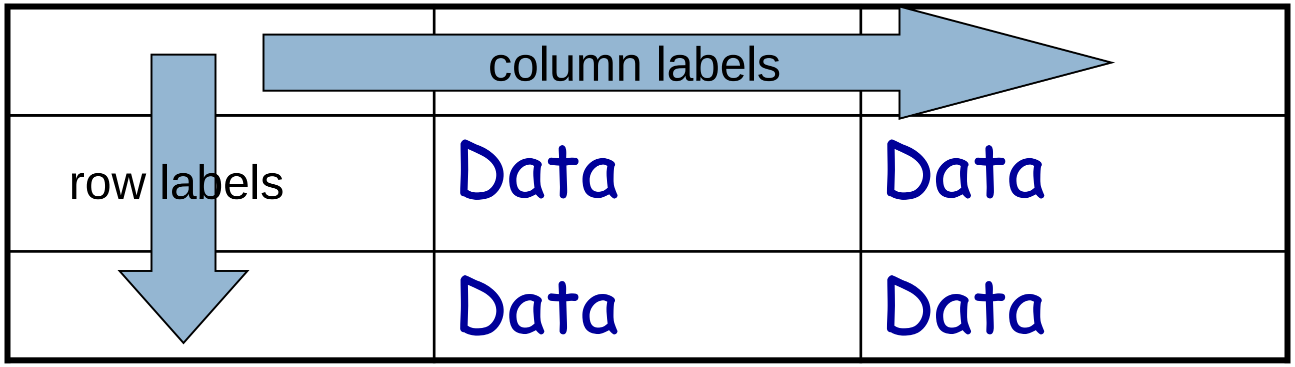


Table Tag

- Specified as content of the block tag <table>
- Border of table is 0 by default
- border attribute is not supported in HTML5
  - must use CSS to specify table border

Table tag

- <caption> tag: gives title to table
- Cells in table are specified one row at a time
  - <tr> tag: specifies each row
  - <td> tag: specifies each cell
  - <th> tag: used to give row/column labels

Table example

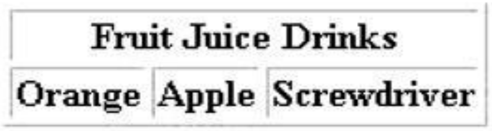
```
<table>
<caption> The 2008 Presidential Election </caption>
<tr>
        <th>Name</th>
        <th>Role</th>
</tr>
<tr>
        <td>Barack Obama</td>
        <td>Democratic nominee for President</td>
</tr>
<tr>
        <td>John Mc. Cain</td>
        <td>Republican nominee for President</td>
</tr>
</table>
```

HTML Code

Example: see table.html

rowspan and colspan

- Use the colspan attribute to specify how many columns to span
  - Use in <th> or <td> tags
- Use rowspan to specify how many rows to span
  - Use in <td> or <th> tag



Fruit Juice Drinks		
Orange	Apple	Screwdriver

```
<tr>
<th colspan="3">Fruit Juice Drinks</th>
</tr>
<tr>
    <th> Orange </th>
    <th> Apple </th>
    <th> Screwdriver </th>
</tr>
```

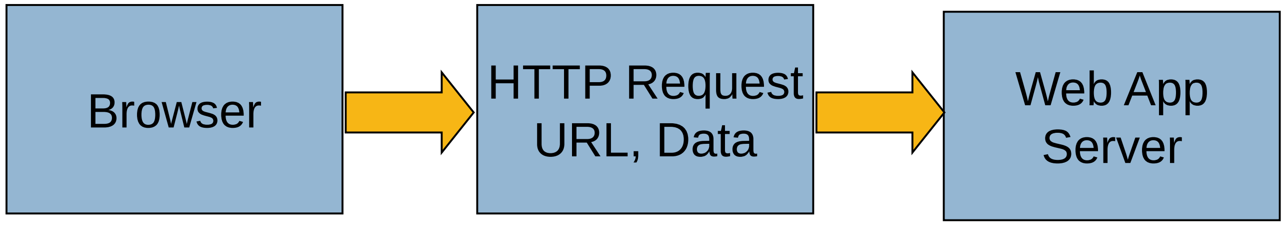
Presentation of Tables

- Adjust how tables look using attributes

Attribute	**Meaning**	**Element**
align	Horizontal placement of data in a cell: **left, right, center**	<tr> <th> <td>
valign	Vertical placement of data in a cell: **top, bottom** and **center**	<th>, <td>
cellpadding	Spacing between the content of a cell and the inner walls of the cell	<table>
cellspacing	Distance between the cells	<table>

Forms

- Communicate data from user to server
- Contains **controls** or **widgets** and labels for controls
- Must have **submit** button to transmit input data to server



<form> tag

- All form components appear in the content of <form> tag
- <form> is a block tag
  - Special type of block tag: can only contain other block tags within it
- Several attributes
  - attribute **action** is required
    - *action* specifies URL of program on the server to be called when user hits *submit* button
  - attribute **method** specifies GET or POST method
    - form data coded into text string

<input> tag

- Inline tag used to specify form controls
  - text, passwords, checkboxes, radio buttons, action buttons, submit, reset, plain
- Required attribute type specifies the kind of control
- All controls other than submit and reset require a name attribute
- Controls checkboxes and radio buttons require an additional value attribute

Simple Form

- HTML Code

```
<form action = “”>
<p>
User name: <input type = “text” name = “Username” size = “25” />
</p>
</form>
```

Example: see simpleform.html

Textbox

- Creates horizontal box for user input
- <input type="text">
  - default size: browser dependent (usually 20 characters)
  - use size attribute to specify size of textbox
    - if user inputs more characters: textbox scrolls
    - if don't want to allow more characters: specify in maxlength attribute

Textbox

- HTML Code

```
<form action = “….” method=”get”>
<p>
Username: <input type = “text” name = “uname” size = “25” maxlength=“25” />
</p>
</form>
```

- Example: see simpleform.html

Password Textbox

- To hide contents of a textbox
- Example: see simpleform.html

```
<form action = "…." method="get">
<p>
Username: <input type = "text" name = "uname" size = "25" maxlength="25" />
Password: <input type = "password" name = "mypass" size = "10" maxlength="10" />
</p>
</form>
```

<label> tag

- HTML code

```
<form action = ”…” method=“get>
<p>
   <label>
     Username: <input type = "text" name = "uname" size = "25" maxlength="25" />
   </label>
</p>
```

- Advantage: text content can be rendered by speech synthesizer

<fieldset> and <legend> tags

- Example: See simpleform.html
- HTML code

```
<form action = ”…" method="get">
<fieldset>
     <legend>Authentication</legend>
      Username: <input type = "text" name = "uname" size = "25”/>
     <br />
     Password: <input type = "password" name = "mypass" size = "10”/>
</fieldset>
</form>
```

Checkboxes

- Used to collect multi-choice input
- Checkbox control
  - single button has two states: checked or not
  - requires a **type**, **name** and a **value** attribute
    - Value of attribute **type** should be **checkbox**
    - all checkboxes in a group must have same value for the **name** attribute
  - attribute **checked** used to specify initial status of checkbox

Checkbox Example

```
<label>
<input type = "checkbox" name = “hobbies" value="swimming" checked="checked" />
Swimming
</label>
<label>
<input type = "checkbox" name = “hobbies" value=“soccer" /> Soccer
</label>
<label>
<input type = "checkbox" name = “hobbies" value=“football" /> American Football
</label>
```

Example: see simpleform.html

Radio buttons

- Also used to collect multi-choice input
- Similar to checkboxes
- Main difference
  - only one radio button can be **on**
- Required attributes: name, type, value
  - Value of attribute type is **radio**
  - all radio buttons in a group must have same value for the name attribute
- checked attribute used to indicate default value

Radio button Example

```
<label>
<input type = “radio" name = “flex" value=“yes" checked="checked" /> Signed up
</label>
<label>
<input type = "radio" name = “flex" value=“no" /> Not signed up
</label>
```

Example: see simpleform.html

Select box

- Used to display menus: compact display of large numbers of options
- <select> tag (rather than <input>)
- Can emulate radio buttons or checkboxes
- name attribute required
- To select more than one choice (ala checkbox): use multiple attribute
- Optional: size attribute

Select box

- Each item in menu specified with <option> tag, nested in <select>
  - optional attribute in <option> tag: selected attribute used to indicate which item is pre-selected

Select example

- HTML code

```
<label> Academic department
   <select name = "dept">
       <option value="IS"> Information Systems </option>
       <option value="CS"> Computer Science </option>
       <option value="BIOL"> Biology </option>
       <option value="CHEM"> Chemistry </option>
   </select>
</label>
```

Example: see simpleform.html

Select example 2

- HTML code

```
<label> Academic department
   <select name = "dept“ size=“2”>
   <option value=“IS”> Information Systems </option>
   <option value=“CS”> Computer Science </option>
   <option value=“BIOL”> Biology </option>
   <option value=“CHEM"> Chemistry </option>
</select>
</label>
```

Action buttons

- Submit and Reset buttons are created with <input> tag
- Role of Submit button
  - form data encoded and sent to server
  - server requested to execute server-resident program specified in action attribute
- Role of Reset button
  - clears all form controls to initial states
- Every form requires a submit button

Example: submit, reset

HTML code

```
<form action = “identify.php”>
<p>
    <input type = “submit” value=“Submit form” />
    <input type = “reset” value=“Reset form” />
</p>
</form>
```

Example: see simpleform.html

Form method: GET vs. POST

- GET passes parameters to server as a query string
  - Limited to browser's URL length, ~100-200 characters
- POST embeds the parameters in HTTP request
  - Not in the URL
- Advantages of POST
  - Information is more private (not shown in browser)
- Disadvantages of POST
  - Can't be bookmarked
  - Browser can't easily go back (POSTDATA error)

Example: Form method and action attributes

HTML code

```
<form method="GET" action = "identify.php">
<p>
    Username: <input type="text" name="uname" />
    Password: <input type="text" name="pswd" />
    <input type = "submit" value="Submit form" />
    <input type = "reset" value="Reset form" />
</p>
</form>
```

Lab: Lists, Tables, Forms

- Open the page you created in previous lab
- Change the line which lists your hobbies into an HTML unordered list
- After the image of your favorite hobby, insert a table with two columns and three rows
  - First column heading: class name
  - Second column heading: class location
- Enter data for two classes to fill up the remaining two rows
- After the table, create a form that takes in
  - your last name
  - your first name,
  - and has a select box with any three options of your choosing,
  - and a submit button
- Save as mypage9.html
- Validate your page and make sure it validates

Organization of elements

- New tags introduced in HTML5
- See organized.html

New elements in HTML5

- <header> tag
  - usually contain one or more <h1-6> tags, logo, authorship information
- <footer> tag
  - usually contain site map links, authorship information, copyright information, etc.
- Both <header> and <footer> tags are block tags
- <header> should not be confused with <head> tag
  - <header> is designed to contain page's headings

<header> tag

- `header` tag
- Example: See organized.html

```
<header>
        <h1> The Podunk Press </h1>
        <h2> "All the news we can fit" </h2>
 </header>
```

<footer> tag

- `footer` tag – a container for footer information

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

Example: See organized.html

Other new HTML5 elements

- The `section` Element – a container for sections. for content that doesn't make sense on its own.
- The `article` Element – a container for self-contained part of a document. for a stand alone piece of content.
- The `aside` Element – a container for tangential info
- The `nav` Element – navigation sections (list of links)

HTML5: The *audio* element

- Prior to HTML5, a plug-in was required to play sound while a document was being displayed
- Audio information is coded into digital streams with encoding algorithms called *audio codecs* – e.g., MP3, Vorbis
- Coded audio data is stored in containers—e.g., Ogg, MP3, and Wav (file name extension indicates the container, not the audio code)
  - Vorbis code is stored in Ogg containers
  - MP3 code is stored in MP3 containers
  - Wav code is stored in Wav containers

The *audio* element

- syntax:

```
  <audio attributes>
     <source src = "filename_1" >
     ...
     <source src = "filename_n" >

   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
- 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
- Example: See audio.html

The *audio* element

- Different browsers have different audio capabilities
- Firefox 3.5+ browsers support Ogg/Vorbis and Wav/Wav
- Chrome 3.0+ support Ogg/Vorbis, MP3/MP3
- IE9+ support MP3/MP3
- Safari 3.0+ support Wav/Wav

The *video* element

- Prior to HTML5, there was no standard way to play video clips while a document was being displayed
- Video information must be digitized into data files before they can be played by algorithms called *video codecs*
- Video codecs are stored in containers
- Video codecs:
  - H.264 (MPEG-4 AVC) – can be stored in MPEG-4
  - Theora – can be stored in any container
  - VP8—can be stored in any container

The *video* element

- Different browsers support different codecs
  - IE9+ support MPEG-4 video containers
  - Firefox 3.5+ support Ogg video containers
  - Firefox 4.0+ support Ogg and WebM containers
  - Chrome 6.0+ support all three
  - Safari 3.0+ support MPEG-4 containers

The *video* element

- Syntax

```
<video attributes>
   <source src = "filename_1" >
   ...
   <source src = "filename_n" >

 Your browser does not support the
audio element
 </video>
```

- Example: See video.html

Attributes of the *video* element

- 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

The *time* element

- For putting a time stamp on a document
- Two parts, text and machine-readable (`datetime`)
  - Machine-readable part `datetime` attribute (optional)
    - Date part: 4-digit year, a dash, 2-digit month, a dash, 2-digit day of the month ("`2012-08-29`")
    - Time (optional) format: `T09:00`
  - Text part is given as the content of `time`
-

```
<time datetime = "2012-08-29T09:00">
     August 8, 2012 9:00 am
  </time>
```

The *time* element

- The two parts need not specify the same date
- Deficiencies:
  - Dates prior to the Christian era are not possible
  - No approximations, cannot specify “circa1900”

Lab: Add new HTML5 elements

- Add the audio and video elements and add an audio and video clip of your choice
- Organize the page you created in the previous lab by using the header and footer elements