Chapter 2: Introduction to HTML Part 1

CS 80: Internet Programming

Instructor: Mark Edmonds

HTML

- What does it stand for?
  - Hypertext markup language
- What is a markup language?
  - Not a traditional programming language
  - Specifies the structure and content of documents

Editing HTML

- All you need is a text editor and a web browser
- Text editors
  - Brackets (recommended), Visual Studio Code, Sublime Text, Notepad++, TextEdit, Vim (!), etc
  - Anything will do, but don't use Microsoft Word (technically you can do this but it will make for a painful semester and you will almost certainly get points off)

Editing HTML

- Web browsers
  - Chrome, firefox, IE, etc
  - For this course, firefox seems to be the most agreeable
    - There is one assignment with Ajax that only works with firefox

HTML Concepts

- Documents are composed of HTML5 *tags*
- These tags outline this structure and content of a webpage
- Tag format:
  - `<tag>` begins the tag
  - `</tag>` closes the tag

HTML Concepts

- Most tags have a beginning and an end
  - Exceptions: <`meta`> (specifies document metadata, <`img`> tag (adds an image to the page), there are others we will come across
  - We call tags without an ending *void elements* because they do not markup text (text is not placed between a start and an end tag).

HTML Concepts

- Tags can be *nested* e.g. there is a tag inside of another tag before the outer tag's closing tag
- Attributes: content inside of a tag that specifies information about this particular use of the tag
- We will see an example in a moment
- Elements: the portions of a HTML document
  - The beginning to closing of a tag form an element
  - e.g. <`tag`> content </`tag`> is an element

Example: `hello_world.html`

- Tip: try clicking the link above! This will load the example in your browser.

```
<!DOCTYPE html>
<!-- document type declaration. required. must be on first line -->

<!-- starts the html document, the root of the document's structure.
    required. -->
<html>

<!-- starts the head section of the document, provides info but usually
     not content. required. -->
<head>
  <!-- metadata tag, here we specify the character encoding of the
       document. required. -->
  <meta charset="utf-8">
  <!-- specify the title of the document (what is displayed in the
       browser tab). required. -->
  <title>Hello world!</title>
<!-- end of head section. required. -->
</head>

<!-- document body. the content goes here! required. -->
```

```html
<body>
  <!-- simple paragraph (the p-tag) -->
  <p>Welcome to HTML5!</p>
<!-- end of body section. required. -->
</body>

<!-- ends the html document. required. -->
</html>
```

HTML5 Validation

- You can validate HTML online!
- Go to https://validator.w3.org/#validate_by_upload to validate a file
- Go to https://validator.w3.org/#validate_by_input to validate input
- Upload `hello_world.html` example to validator

Headings

- Heading elements designate a level of importance for a topic on a page
- Headings range from `<h1>` to `<h6>`
  - The lower the number, the greater importance
  - `<h1>` is the "most" important

Example: `headings.html`

```html
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>Headings</title>
</head>

<body>
  <h1>Level 1 Heading</h1>
  <h2>Level 2 heading</h2>
  <h3>Level 3 heading</h3>
  <h4>Level 4 heading</h4>
  <h5>Level 5 heading</h5>
  <h6>Level 6 heading</h6>
```

```html
</body>

</html>
```

Hyperlinks

- Provides a link to another HTML document
- Can be on this host or on a different host
  - What does this mean?
  - You can link internally to your own content or to a new host's webpage.
  - In either case, a new HTTP request is triggered

Hyperlinks

- Links are facilitated using the `<a>` tag with a corresponding attribute of `href`
  - `a` stands for 'anchor'
  - `href` stands for 'hypertext reference'
- We can href other protocols (e.g. `https://`, `ftp://`, `mailto:`, `file:`, etc) or even javascript!
  - If it's javascript the script will execute when clicked
  - This will make more sense once we get to javascript, hold on!

Example: `linking.html`

```html
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>Linking</title>
</head>

<body>
  <h1>Welcome to CS 80 at SMC</h1>

  <p>Hope you enjoy; make sure to <strong>turn in assignments on time</
      strong></p>

  <p><a href="http://www.smc.edu/">SMC</a></p>
  <p><a href="http://smconline.org/index.real?action=Login">eCompanion
      </a></p>
  <p><a href="mailto:edmonds_mark@smc.edu">Email me</a></p>
</body>

</html>
```

Images

- Added to HTML using the `<img>` tag
- `<img>` tag is one of the tags without an ending `</img>`
- The `src` attribute specifies where the image is located
  - The location can be a relative path (e.g. stored on the same computer as the html document)
  - The location can be a remote path (e.g. an image stored on a different host)

Images

- Must use an `alt` attribute, it is allows for two important usages:
  - For those with poor or no eyesight to still understand the content on the page. A text-to-speech program can read the `alt` description to a visually impaired person.
  - If the image fails to load (maybe it's an external image), the `alt` can still describe what the image is
  - The `alt` description should be as brief as possible while still being descriptive

Images

- Common attributes used: specifying height and width'
  - Height and width are both measured in pixels
  - If there are no height and width are specified, image will be rendered at it's own size (the original image size)
- Images can be nested inside of a link tag (`<a>`) to create an image that is also a hyperlink

Example: `images.html`

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>Images</title>
```

```html
</head>

<body>
  <h3>Every common image type is supported</h3>

  <p><strong>Note: alt is a required attribute</strong></p>
  <p>
  <!-- relative path -->
  <img src="./camel.png" alt="a camel" >
  <!-- remote paths -->
  <img src="https://cdn.meme.am/instances/500x/37834819.jpg" alt="The
     woes of programming" >
  <img src="http://www.lakeshorebranding.com/wp-content/uploads
     /2011/12/web-design-chicago-438x275.png" alt="The woes of
     programming" >
  </p>

  <h3>We can control the width and height as well</h3>

  <img src="https://media.giphy.com/media/eCqFYAVjjDksg/giphy.gif" alt=
     "Hacking in progress" height="200" width="350">

  <h2>We can make our images links as well!</h2>

  <a href="http://google.com">
    <img src="https://cdn.vox-cdn.com/thumbor/YodYS9ma7P_8jcAplZSoIlw4v
       -c=/0x0:2012x1341/920x613/filters:focal(0x0:2012x1341)/cdn.vox-
       cdn.com/uploads/chorus_image/image/47070706/google2.0.0.jpg" alt
       ="The google logo."/>
  </a>
</body>

</html>
```

Special Characters

- HTML itself uses certain characters to represent the structure of the document
  - These symbols have special meaning within the document itself
  - What are some examples?
    - How are tags wrapped?
      - Around < and >

Special Characters

- Clearly, we still want to be able to include these tags in an HTML document.
- Solution: We have to use a special convention to include these restricted characters
- These special conventions are called *character entity references*
  - The basic format: `&ref;`
    * Where `ref` is a reference to the character you wish to insert

Special Characters

- E.g. `&lt;` inserts the less-than symbol (<)
- You can find a list of these symbols online
  - Do a quick google search go all of the symbols available
  - http://www.w3schools.com/html/html_entities.asp
  - https://www.w3.org/TR/REC-html40/sgml/entities.html
  - https://dev.w3.org/html5/html-author/charref

Example: `char.html`

```html
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>Special characters and horizontal rules</title>
</head>

<body>
  <p>
    <!-- use some special characters -->
    We &amp;amp; can &lt; add &copy; symbols &trade; whereever &hellip;
        we &quot; want &gt; in &mdash; our &ndash; docment
  </p>
  <hr>
  <p>
    It's best practice to avoid the &lt;hr&gt; tag now because we can
       do styling with CSS, producing more professional looking and
       easier to maintain websites
    We can also strike through <del>text</del>, subscript <sub>text</
       sub>, or superscript <sup>text</sup>.
  </p>
```

```
</body>

</html>
```

Lists

- Two types of lists:
  - Unordered list and ordered list
- Unordered list (bulleted list; like this list)
  - `ul` tag starts an unordered list (`ul` stands for unordered list)
  - Each list item is a nested `li` tag (`li` stands for list element)

Lists

- Ordered list (numbered list)
  - `ol` tag starts an ordered list
  - each list item is a nested `li` tag
- We can also make lists inside of lists (nested lists)
  - Just like your word processor
  - Bullets of an unordered list will change based on the nesting level

Example: `lists.html`

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>Lists</title>
</head>

<body>
  <h2>We can make unordered and ordered lists!</h2>

  <p>
    This is an unordered list
  </p>
  <ul>
    <li>this item has no order</li>
    <li>neither does this one</li>
    <li>this is not an exciting list</li>
  </ul>

  <p>
    This is an ordered list with reasons why this course is good
  </p>
  <ol>
    <li>Computer science is a great field</li>
    <li>The course is fun!</li>
    <li>The tests are take home</li>
  </ol>

  <p>
    We can nest lists too
  </p>
  <ol>
    <li>Computer science is a great field
      <ul>
        <li>It's fun</li>
        <li>There are a lot of jobs</li>
        <li>It makes you think</li>
      </ul>
    </li>
    <li>The course is fun!
      <ol>
        <li>You get to explore an entire subfield in one semester
          <ul>
            <li>Can be the starting point for a career</li>
          </ul>
        </li>
      </ol>
    </li>
    <li>The tests are take home
    <ul>
      <li>
        A nest
        <ul>
          <li>
            Another nest
            <ul>
              <li>Nests on nests</li>
            </ul>
```

```
                </li>
            </ul>
        </li>
    </ul>
</ol>


</body>

</html>
```

Line Breaks

- <br> tag - a line break

Tables

- Similar to a textbook table or excel spreadsheet (except without math functions)
- Started with the `table` tag
- The nested `caption` tag gives the table a title and summarizes the table's content

Tables

Table Bodies, Headers, and Footers

- <thead> and <tfoot> specify the header and footer of the table, respectively
  - Maybe you want different styling for the header and footer using CSS
  - Again, we'll cover that later

Tables

Table Bodies, Headers, and Footers

- <tbody> specifies the main body portion of the table
- <thead> and <tfoot> have the same internal structure

Tables

Table Rows

- `<tbody>`, `<thead>`, and `<tfoot>` is comprised of at least one `<tr>` element, which is a table row
  - Note that no table requires all three, but you should use them appropriately
- If your table has a header, put the header, etc.
- Makes table maintenance easier later

Tables

Table Rows

- Each `<tr>` tag is comprised of `<th>` tags, which is a *header cell*. It is different from the normal cell!
  - It is different to allow easier styling using CSS
  - The table fills left to right along the columns
- Repeat this `<tr>` and `<th>` pattern for each row, and each column you want to specify
- `<tfoot>` can be below or above

Tables

Table Rows

- `<tbody>` follows the same convention, but uses a `<td>` element instead of `<th>`
- `td` stands for 'table data'

Tables

Table Sizing

- By default, each table column is only as wide as its largest cell.
- `rowspan` and `colspan`
- These attributes allows a cell to span multiple rows or columns
  - This is like merging rows/columns in a document

Tables

Table Sizing

- `rowspan` allows a single table cell to span the width of more than one cell or row
- `colspan` allows a single table cell to span the width of more than one cell or column
- These attributes can be applied in `<th>` and `<td>` elements

Example: `tables.html`

```html
<!DOCTYPE html>
<html>
<!-- Fig. 2.12: table1.html -->
<!-- Creating a basic table. -->

<head>
  <meta charset="utf-8">
  <title>Tables</title>
</head>

<body>

  <h2>Here's a simple table example</h2>

  <h3>Notice the table scales to the widest cell's content (and not the
      caption's)</h3>

  <!-- the <table> tag opens a table -->
  <!-- Reminder: the border element should not be used. We are using it
      here before we learn CSS -->
  <table border="1">
    <!-- the <caption> tag summarizes the table's -->
    <!-- contents (this helps visually impaired people) -->
    <caption><strong>Table of Fruits (1st column) and Their Prices (2nd
        column)</strong></caption>
    <!-- the <thead> section appears first in the table -->
    <!-- it formats the table header area -->
    <thead>
      <tr>
        <!-- <tr> inserts a table row -->
        <th>Fruit</th>
        <!-- insert a heading cell -->
        <th>Price</th>
      </tr>
    </thead>
    <!-- the <tfoot> section appears last in the table -->
    <!-- it formats the table footer -->
    <tfoot>
      <tr>
        <th>Total</th>
        <th>$3.75</th>
      </tr>
    </tfoot>
    <!-- within the <tbody> -->
    <tbody>
      <tr>
        <td>Apple</td>
        <!-- insert a data cell -->
        <td>$0.25</td>
      </tr>
      <tr>
        <td>Orange</td>
        <td>$0.50</td>
      </tr>
      <tr>
        <td>Banana</td>
        <td>$1.00</td>
      </tr>
      <tr>
        <td>Pineapple</td>
        <td>$2.00</td>
      </tr>
    </tbody>
  </table>

  <!-- <br> is a line break -->
  <br>

  <h2>We can make more complicated tables using rowspan and colspan</h2>

  <!-- Reminder: the border element should not be used. We are using it here before we learn CSS -->
  <table border="1">
    <caption>A more complex sample table</caption>
    <thead>
      <!-- rowspans and colspans merge the specified -->
      <!-- number of cells vertically or horizontally -->
      <tr>
        <!-- merge two rows -->
        <th rowspan="2">
          <img src="camel.png" width="205" height="167" alt="Picture of a camel">
        </th>
```

```
        <!-- merge four columns -->
        <th colspan="4">
          <strong>Camelid comparison</strong><br> Approximate as of
              10/2011
        </th>
      </tr>
      <tr>
        <th># of humps</th>
        <th>Indigenous region</th>
        <th>Spits?</th>
        <th>Produces wool?</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Camels (bactrian)</td>
        <td>2</td>
        <td>Africa/Asia</td>
        <td>Yes</td>
        <td>Yes</td>
      </tr>
      <tr>
        <th>Llamas</th>
        <td>1</td>
        <td>Andes Mountains</td>
        <td>Yes</td>
        <td>Yes</td>
      </tr>
    </tbody>
  </table>
</body>

</html>
```

Forms

- Mechanism for the user to send data to the server from the client
- The user types in the data, then hits 'submit' the submit triggers a
- HTTP request to the server to accept the form
  - The receiving end will be covered in Chapter 17
- Forms use the `<form>` tag

Forms

Important `<form>` attributes

- These setup the form and specify how and where to send data to the server
- `method` attribute can be two values:
  - `get`
  - `post`

Forms

Important `<form>` attributes

- `get`
  - Appends form data to url in name/value pairs
  - Length of URL is limited (~3000 characters limit)
  - Never use this to send sensitive data
  - Best suited for things like a query or other non-secure data (you can bookmark the url to effectively save the form submission)

Forms

Important `<form>` attributes

- `post`
  - Appends form data to HTTP request
  - Has no size limitations
  - Cannot be bookmarked (since it does not modify the URL)

Forms

Important `<form>` attributes

- `action`
  - Specifies where to send the form data (*e.g.* what site should process the form)
  - Must be a valid URL
  - The server **should** know how to respond to the form submission

Forms

Inputs

- Once we have the form setup, we can add inputs
- Inputs are added using the `<input>` tag

Forms

Important `<input>` attributes

- `type`
  - Controls the type of the input; a lot of options available
  - Options include: `text`, `button`, `color`, `password`, `radio`, `range`, `reset`, `submit`, etc. with more available online

Forms

Important `<input>` attributes

- `type`
  - `hidden` is a special type
    * It submits data to the server that is predetermined in the HTML page
    * The user cannot control this input
    * Might be used for sending information to another server to identify where the form is coming from

Forms

Important `<input>` attributes

- `type`
  - `text` has a couple of special attributes: `size` - specifies the size of the text box and `maxlength` which specifies the maximum length of the input

Forms

Important `<input>` attributes

- `name`
  - Gives the input a name that can be referenced once the server receives the submission
  - This is part of the glue between the client and the server that processes the form

Forms

Important `<input>` attributes

- `value`
  - Gives the input an initial value

Forms

- Please lookup these elements in your book or online:
  - `textarea` - multiline text input, has rows and cols attributes
  - `password` - provides a password-protected field. this is only visually enforced (displays a * instead of the text), the password still should be encrypted when sent over HTTP
  - `color` - allows color input
  - `number` - allows user to input a number, similar to a text but for numbers

Forms

- Please lookup these elements in your book or online:
  - `range` - allows user to pick between a range of values
  - `checkbox` - allows user to tick multiple options
  - `radio` - allows user to pick one option from a list
  - They all follow a similar pattern, but familiarize yourself!
- `<select>` tag presents a dropdown menu with a preselected list of options

Example: `forms.html`

```html
<!DOCTYPE html>
<html>

<!-- Fig. 2.14: form.html -->
<!-- Form with a text field and hidden fields. -->

<head>
  <meta charset="utf-8">
  <title>Feedback Form</title>
</head>

<body>
  <h1>Feedback Form</h1>
  <p>Please fill out this form to help us improve our site.</p>
```

```html
    <!-- this tag starts the the form, gives the -->
    <!-- method of sending information and the -->
    <!-- location of the form-processing script -->
    <form method="post" action="http://www.deitel.com">
      <!-- hidden inputs contain non-visual -->
      <!-- information that will also be submitted -->
      <input type="hidden" name="recipient" value="deitel@deitel.com">
      <input type="hidden" name="subject" value="Feedback Form">
      <input type="hidden" name="redirect" value="main.html">
      <!-- <input type = "text"> inserts a text field -->
      <p>
        <label>Name:
          <input name = "name" type = "text" size = "25" maxlength = "30"
              >
        </label>
      </p>

      <p>
        <label>Comments:<br>
          <textarea name = "comments" rows = "4" cols = "36">Enter
              comments here.</textarea>
        </label>
      </p>
      <p>
        <!-- input types "submit" and "reset" insert -->
        <!-- buttons for submitting and clearing the -->
        <!-- form's contents, respectively -->
        <input type="submit" value="Submit">
        <input type="reset" value="Clear">
      </p>
    </form>

  </body>

  </html>
```

Internal Linking

- Mechanism to jump between locations in a single document without reloading the HTML page
- Basic idea: we uniquely mark elements in the document using the `id` attribute, then we refer to the corresponding id in an anchor (a link, the `a` tag)

- The link tag can reference a specific tag in a different HTML document, even on a different host

Example: `internal_linking.html`

```html
<!DOCTYPE html>
<html>
<!-- Fig. 2.16: internal.html -->
<!-- Internal Linking -->
<head>
  <meta charset="utf-8">
  <title>Internal Linking</title>
</head>

<body>
  <!-- id attribute creates an internal hyperlink destination -->
  <h1 id="features">The Best Features of the Internet</h1>
  <!-- an internal link's address is "#id" -->
  <p><a href="#bugs">Go to <em>Favorite Bugs</em></a></p>
  <ul>
    <li>You can meet people from countries around the world.</li>
    <li>You have access to new media as it becomes public:
      <ul>
        <li>New games</li>
        <li>New applications
          <ul>
            <li>For Business</li>
            <li>For Pleasure</li>
          </ul>
        </li>
        <li>Around the clock news</li>
        <li>Search Engines</li>
        <li>Shopping</li>
        <li>Programming
          <ul>
            <li>HTML5</li>
            <li>Java</li>
            <li>Dynamic HTML</li>
            <li>Scripts</li>
            <li>New languages</li>
          </ul>
        </li>
      </ul>
    </li>
    <li>Links</li>
    <li>Keeping in touch with old friends</li>
    <li>It is the technology of the future!</li>
  </ul>

  <br><br><br><br><p>Skipping a whole bunch of space where you would put amazing web content</p><br><br><br><br>
  <p>Skipping a whole bunch of space where you would put amazing web content</p><br><br><br><br>
  <p>Skipping a whole bunch of space where you would put amazing web content</p><br><br><br><br>
  <p>Skipping a whole bunch of space where you would put amazing web content</p><br><br><br><br>
  <p>Skipping a whole bunch of space where you would put amazing web content</p><br><br><br><br>

  <!-- id attribute creates an internal hyperlink destination -->
  <h1 id="bugs">My 3 Favorite Bugs</h1>
  <p>
    <!-- internal hyperlink to features -->
    <a href="#features">Go to <em>Favorite Features</em></a>
  </p>
  <ol>
    <li>Fire Fly</li>
    <li>Gal Ant</li>
    <li>Roman Tic</li>
  </ol>

  <h1>We can even reference an id in a different HTML document on a different host</h1>
  <p>
    <a href="https://en.wikipedia.org/wiki/HTML#Attributes">https://en.wikipedia.org/wiki/HTML#Attributes</a>
  </p>
  <p>
    <a href="#">Top</a>
  </p>

</body>

</html>
```