CSCE 120: Learning To Code

Module 5.0: Presenting Data I

Introduction

This module introduces you to Hypertext Markup Language (HTML) and the Document Object Model (DOM).

HTML Overview

Hypertext Markup Language (HTML) is the standard *markup* language used on the web and in web pages. A markup language is a language used in a document that is able to specify both *content* and *semantics*.

In a text document, the reader is able to recognize which collection of words form a sentence, which collections of sentences form a paragraph, etc. They would also be able to easily identify the meaning of words either directly or through context. For example, they would easily be able to identify which words represented names or locations, etc.

A computer, however, isn't that smart. Instead, we must use a markup language to "tag" certain elements with a semantic meaning. For example, we could begin and end each paragraph with special beginning `<p>` and ending `</p>` tags to "mark" it as a paragraph. When displayed to a user these tags would be redundant (and are usually *not* displayed in a rendered document). However, they are invaluable to a computer to be able to process, render, and display a document.

HTML Basics

Every HTML document has the same basic elements. Since HTML5, every HTML document should begin with a Document Definition Tag: `<!DOCTYPE html>` which indicates that the file should be interpreted as an HTML5 document. Historically, this tag is used to tell the browser which version of HTML the document conforms to.

After this opening tag, each HTML *element* is denoted with an opening and closing tag. The document content begins with the opening `<html>` tag and ends with the closing `</html>` tag. This element is known as the *root* element of the document (similar to the root element in JSON data). All other document elements are placed inside these tags.

Commonly, documents are split into two parts: the head and the body using the tags `<head>...</head>` and `<body>...</body>` respectively. The main content of the document is placed into the body. The head usually contains information about the document instead referred to as *metadata*. In particular, the `<title>` tag is used to denote the title of the document. In addition, external resources such as style sheets an scripts are often placed in the head. Other metadata such as keywords or other items can be placed in the head to make it easier for search services such as Google to *index* your page and content for better search results. The process of defining proper metadata is often part of a Search Engine Optimization (SEO).

As with JSON, HTML tags must be *well-balanced*: every opening tag must have a corresponding closing tag and they must be properly nested. We can also include comments in HTML using the opening `<!--` tag and closing `-->` tag. Everything between these two tags is treated as a comment whether its on a single line or spans multiple lines. Comments in HTML are ignored and have no effect on the document or its rendering.

Finally, just as some special characters in JavaScript strings must be *escaped*, certain characters in HTML should be escaped to be displayed properly. There are 5 characters that *must* be escaped (see Table 1). Other characters can be escaped, but will usually be displayed properly even if they are not. A complete list can be found here: http://www.theukwebdesigncompany.com/articles/entity-escape-characters.php

Character	HTML
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&#39;

Table 1: Special Characters in HTML

Here is a minimal HTML document:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World!</title>
  </head>
  <body>
  <!-- rest of the body content here -->
  </body>
</html>
```

Tags

The HTML specification defines dozens of tags. For a complete reference, see http://www.w3schools.com/tags/. However, we will only highlight a few of the more useful ones here.

Headings

Major sections of a document can be denoted using a heading tag. There are 6 levels of headings using tags `<h1>` through `<h6>`. An example:

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
```

A rendering of this example in Figure 1.

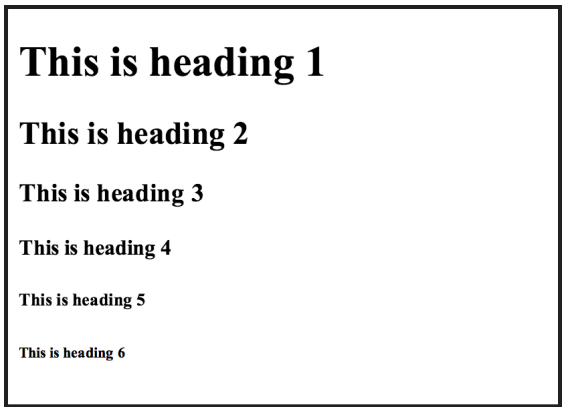

Figure 1: Rendering of Heading Examples

Historically, the most common use of these tags has been to denote titles of sections, subsections, and subsections of subsections, etc. HTML5 introduced the `<section>` tag which can be nested to better define sections/subsections. However, the heading tags are still commonly used and supported.

Divisions

Sections or areas of a document can be denoted using the division tag, `<div>`. By default, simply using the tags has little effect on the rendered document. Divisions are used to group elements together into one logical area or section of the rendered page.

For this reason, division elements are often nested. Divisions are often used to define the *layout* of a web page. We will explore this in depth later on.

Paragraphs

Paragraphs in HTML are identified with the `<p>` tag. The most common default rendering of a paragraph is to add some extra vertical space between paragraphs. Note that the HTML specification does *not* allow you to nest paragraph elements (it does not make sense that a paragraph contains a paragraph after all).

Lists

Bullet pointed and numbered lists can be included using the `<ul>` (unordered list) and `<ol>` (ordered-list) tags. Items in the list can be specified using the `<li>` (list-item) tag. Lists can be nested which usually changes the style of the bullets or a change in the numbers used in an ordered list.

```
<ul>
  <li>Item A</li>
  <ul>
    <li>Sub Item a</li>
    <li>Sub Item b</li>
  </ul>
  <li>Item B</li>
  <li>Item C</li>
</ul>
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
</ol>
```

A rendering of this example in Figure 2.

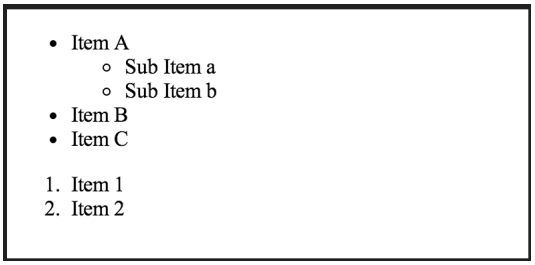

Figure 2: Rendering of List Examples

Tables

Tables are a common element of many documents. Often they are used to display data or organize records. In HTML there are several tags associated with creating a proper table. The first tag for any table is the `<table>` tag. Like the HTML document itself, a table can have both a head and a body using the tags `<thead>` and `<tbody>` respectively. The head is typically used for labeling the columns while the body is used to contain the actual records/data. Entries in the table are in a row-first manner: that is, we define a row, then within the row we specify column records, called *cells*. Each row is indicating using the `<tr>` tag (table-row), while each cell is indicated using the `<td>` tag (table-data). In the case of a header row, we should alternatively use the `<th>` tag (table-header).

It is important to ensure that the number of cells in each row is consistent and the same across each row. It is also necessary to ensure that all tags are well-balanced and follow the expected row-first formatting. Failure to properly define a table may result in an oddly rendered web page. A full example follows.

```
<table>
  <thead>
    <tr>
      <th>Game Title</th>
      <th>System</th>
      <th>Release Date</th>
    </tr>
  </thead>
  <tbody>
  <tr>
    <td>The Legend of Zelda</td>
    <td>Nintendo Entertainment System</td>
    <td>1986-02-21</td>
  </tr>
  <tr>
    <td>Katamari Damacy </td>
    <td>PlayStation 2</td>
    <td>2004-03-18</td>
  </tr>
  <tr>
    <td>2048</td>
    <td>Mobile</td>
    <td>2014-03-09</td>
  </tr>
  </tbody>
</table>
```

A rendering of this example in Figure 3.

Game Title	System	Release Date
The Legend of Zelda	Nintendo Entertainment System	1986-02-21
Katamari Damacy	PlayStation 2	2004-03-18
2048	Mobile	2014-03-09

Figure 3: Rendering of Table Examples

A very old way of designing web page layouts was to use tables to define sections. Do not be tempted to do this. There are much better ways to define modern, responsive layouts that work on multiple platforms (mobile, web browser, etc.).

Anchors & Links

The most common element associated with web pages is the hyperlink: a clickable reference to another page or resource. Links are actually specified with an *anchor* tag using the tag `<a>`. The idea behind an anchor element was originally much more general: it was a generic reference to another resource, including resources within the same page.

The typical usage, however, is to provide a hyperlink to another page. To specify that resource, we need to use an *attribute*. We'll examine attributes in depth; for now, we simply highlight the usage of the `href` attribute. An example:
`<a href="http://cse.unl.edu">CSE Home Page</a>`. Note the syntax: within the opening `<a>` tag, we added a key-value pair (similar to the key-value pairs in JSON, but with different syntax). In particular, the `href` is the key and `"http://cse.unl.edu"` is the value, a string denoted with double quotes. When the web page is rendered, this element will appear as a clickable link with the words CSE Home Page. We can change the page or resource it points to by changing the value of the `href` and we can change the label or wording used in the rendering by changing the text inside the anchor tag.

Further, we used the protocol label `http://` in the address. This is usually used for links to other pages hosted on other servers. You can make references to pages and resources on the same server by omitting the protocol prefix and instead providing a *relative* reference. As an example, suppose your page is stored in a directory named `myHome` and in that directory you have another directory named `myStuff`. You can refer to a resource stored in the directory above `myHome` using something like:

```
<a href="../index.html">Go to home page</a>
```

Here, `..` is used to indicate "one directory above". Directories are separated using the forward slash. You can make reference to a resource in the subdirectory by using something like:

```
<a href="myStuff/index.html">Go to home page</a>
```

Images

Images can be embedded in a web page using the `<img>` tag. Like the anchor tag, we have to specify where the image source (file) is located using a `src` (source) attribute. An example:

```
<img src="./myPhoto.jpg"/>
```

The `./` indicates that the image file is located in the *same* directory as the current page and in general, can be omitted. Another thing to note is that there was no text associated with this tag as it represented an image. We could have written it as

```
<img src="./myPhoto.jpg"></img>
```

but to shorten this up, we wrote the forward slash at the end of the opening tag, eliminating the need for a closing tag. This is known as an *empty element*. Other empty elements that do not contain inner elements or text can be written similarly.

Forms

Many web pages interact with a user by allowing them to fill out forms to provide input data. Most of the time, the input data is then submitted to a server which then sends a response (usually a new page). Our focus will be on the *client side* (that is, the browser) so we won't generally be submitting form data to a server, but instead retrieving and using form data in JavaScript code.

A form is defined using the `<form>` tag. Inside the form, several pieces of data can be defined of various forms. The most common form elements are:

- A text input – a small text box that allows the user to type in information
- Radio buttons – clickable buttons where a set of selections can be made, but typically the user is only allowed to select one
- Check boxes – clickable boxes that can be selected or unselected
- Drop downs – A predefined list of options that the user can select from by clicking a drop down menu that expands when clicked. There are also ways to allow the user to make multiple selections from the list.

There are several more form elements that can be used, a complete list can be found here: http://www.w3schools.com/html/html_forms.asp, but we'll focus on those listed above. The first three can be defined using the `<input>` tag. To distinguish the type of input, you use a `type` attribute which can be set to one of the following: `text`, `radio`, or `checkbox`. To associate different choices for radio buttons and checkboxes you use another attribute inputs with each other, you use the `name` attribute and use the same name value for all options in the same group. The `value` attribute is used to define a value for that selection. A drop down input is defined using the `<select>` tag with

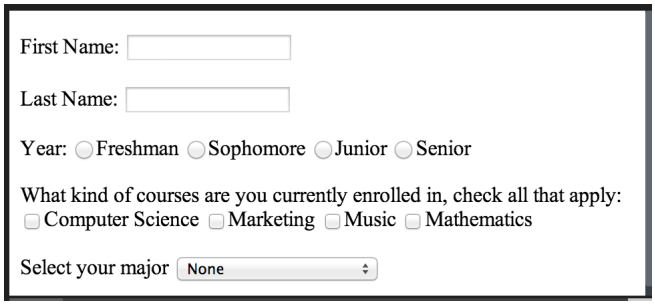

Figure 4: Rendering of Form Examples

nested `<option>` elements corresponding to each element in the menu.

A full example:

```
<form>
  <p>First Name: <input type="text" id="firstName" value=""/></p>
  <p>Last Name: <input type="text" id="lastName" value=""/></p>
  <p>Year:
   <input type="radio" name="year" value="1">Freshman</input>
   <input type="radio" name="year" value="2">Sophomore</input>
   <input type="radio" name="year" value="3">Junior</input>
   <input type="radio" name="year" value="4">Senior</input>
  </p>
  <p>What kind of courses are you currently enrolled in, check all that apply:
    <input type="checkbox" name="courseType" value="CSCE">Computer Science</input>
    <input type="checkbox" name="courseType" value="MRKT">Marketing</input>
    <input type="checkbox" name="courseType" value="MUNM">Music</input>
    <input type="checkbox" name="courseType" value="MATH">Mathematics</input>
  </p>
  <p>Select your major
    <select id="club">
      <option value="none">None</option>
      <option value="CENG">Computer Engineering</option>
      <option value="MNGT">Management</option>
      <option value="BSED">Education</option>
    </select>
  </p>
</form>
```

A rendering of this example in Figure 4.

Buttons can be added using the `<button>` tag, `<button>Compute!</button>` which will render the button with the text "Compute!" on it. Later on we'll see how to add functionality to buttons.

Tag Attributes

As we've previously seen, elements can have content (the stuff that appears between the opening and closing tags which may include text or other HTML elements), but they

can also have *attributes*. These are key-value pairs that are placed inside the opening tag. An element can be given any attribute, but only some are recognized by the HTML standard. Multiple attributes are separated by spaces while the key is written as a string (no spaces, lower casing is recommended, but attribute names are case insensitive). The value for the attribute is specified using the equal sign followed by a string encapsulated with double quotes.

The official HTML specification defines certain attribute names that are valid for certain elements, though other frameworks or applications may define their own attributes. Two of the most important attributes that we'll address in the next topic are `style` and `class` which are used to define how the element is ultimately rendered in the browser. Everything from the color of the text, to its orientation on the page can be specified using Cascading Style Sheets (CSS) through these two attributes.

Document Object Model

HTML documents are highly structured. HTML elements can contain other HTML elements if they are placed within the opening and closing tags. This naturally defines a *hierarchy* for the document: the `<html>` element is the *root* of the tree that "owns" all other elements. This is often depicted as a *tree* data structure.

Trees in Computer Science consist of *nodes* which are connected to each other by *edges*. They are often depicted with the root node at the top and *child* nodes oriented below. As an example, consider the HTML document below and the corresponding document tree depicted in Figure 6. The `<html>` node has two children, the `<head>` node and `<body>` node. Each of those nodes owns children, etc.

This type of organization is called the *Document Object Model* or DOM. Using JavaScript we can easily interact and manipulate objects in the DOM representation. For example, we can apply changes to a particular node and all of its children, or to all nodes of a certain type (element tag), or to all nodes with a certain *class*.

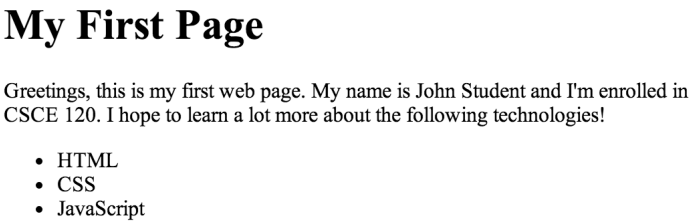

Figure 5: Rendering of Full Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World!</title>
    <link rel="stylesheet" type="text/css" href="jquery-ui.css" />
    <script type="text/javascript" src="jquery.min.js"></script>
  </head>
  <body>
    <h1>My First Page</h1>
    <p>Greetings, this is my first web page.  My name is John
    Student and I'm enrolled in CSCE 120.  I hope to learn a lot
    more about the following technologies!</p>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
  </body>
</html>
```

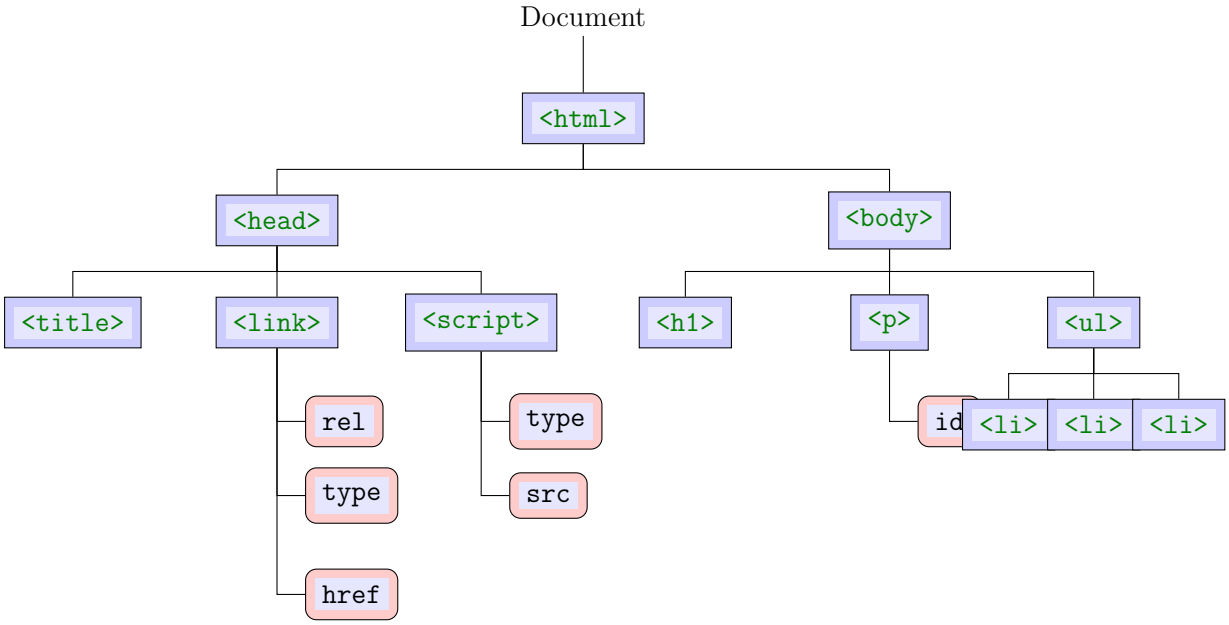

Figure 6: Document Object Model Tree. Element values have been omitted for clarity.