HTML Notes

HTML BASICS & SYNTAX:

HTML is responsible for structuring the **content** It's a markup language, not a programming language. There are no control structures or logic--just a bunch of tags to memorize. Your web browser is responsible for taking an HTML document (which is what is actually returned by servers) and rendering it as a webpage.

TAGS:

HTML is made up of a bunch of "tags" within angle brackets. They look like this:

```
<html> Stuff! </html>
```

HTML tags can also have optional attributes attached to them, like this:

```
<html class="foo"> Stuff! </html>
```

Some tags need to be closed, and some don't. Unpaired elements work with only an opening tag:

```
<p>A bunch of random text here!</p> <!-- I'm a paragraph tag, I need to be closed! -->
<img src="myImage.png" alt="My Image!" /> <!-- I'm a IAMGE tag, I don't need to be closed! -->
```

The purpose of tags is to **semantically represent** what's inside them, if possible. This is for your sanity, as well as for search engines (though it matters less now than it did). There are many, many tags available.

A (VERY) BASIC HTML PAGE:

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Placeholder Title</title>
    </head>
    <body>

    </body>
</html>
```

Alright, so line-by-line:

`<!DOCTYPE html>`

This is what’s referred to as a doctype declaration. It tells the browser what kind of document you have. The declaration seen here is an HTML5 declaration. Older doctypes used to look a lot worse, like this:

```html
<!DOCTYPE HTML
          PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
```

`<html> </html>`

This tag specifies an html document. Everything in your document (except you doctype) goes inside this.

`<head> </head>`

This tag contains the *non-displayable* parts of your web page. There are many, many things that can go here, but we'll see those a little bit later.

`<body> </body>`

This is where everything *displayable* goes. The browser will attempt to display everything you put inside this tag.

COMMON TAGS:

`<a>`

Creates a hyperlink to the resource specified in the href attribute. For example:

```
<a href="http://www.example.com">I'm a hyperlink!</a>
```

`<p>`

Denotes a pargraph.

`<h1>, <h2>, <h3>, <h4>, <h5>, <h6>`

Denotes a header. **`<h1>`** represents the most important or prominent header, and **`<h6>`** the least prominent or important. You likely won’t need to go all the way to **`<h6>`** for the majority of your projects.

`<img>`

Displays an image. For example:

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

`<div>`

An element that has no semantic meaning. Used primarily as a “container” when no semantically correct tag exists. This is a **block-level element** (more on this in the CSS Notes).

`<span>`

Another element that has no semantic meaning. Serves much the same role as
, but is in an **inline element** (again, more on this in the CSS section).

`<ul>`

An un-ordered list. Contains one or more **`<li>`** (list item). By default, the browser will render each **`<li>`** within the **`<ul>`** as a bullet point. For example:

```
<ul>
    <li> Item One </li>
    <li> Item Two </li>
    <li> Item Three </li>
</ul>
```

`<ol>`
The same as un-ordered list, but ordered! By default, the browser will automatically number all **`<li>`**'s inside an **`<ol>`**.

`<iframe>`

Used to display a webpage within a webpage (Inception webpage!). Frames are generally considered bad practice (in fact, all other kinds of frames have been disallowed in HTML5), but necessary for some things. Facebook "Like" buttons are done using iframes. Example:

```
<iframe src="URL"></iframe>
```

`<!-- this is a comment -->`

Used to place comments directly in the source code

TABLES:

You should use them for tabular data, and that's it! They are not a replacement for actually learning how to do proper layout in CSS (more on that later)! In any case, if you absolutely must use a table, here's an example from w3:

```
<table>
    <tr> <!-- a table row -->
        <td>row 1, cell 1</td> <!-- a table data cell -->
        <td>row 1, cell 2</td>
    </tr>
    <tr>
        <td>row 2, cell 1</td>
        <td>row 2, cell 2</td>
    </tr>
</table>
```

IDS AND CLASSES

Perhaps the most common attributes on HTML tags are “id” and “class”. These attributes allow you to access HTML elements in external resources like stylesheets or scripts. The main difference is that **IDs** should only be used **once per page**, while **Classes** should be used **multiple times per page**. Note that this *isn’t* enforced by the browser. Your page will likely still render fine if you have two elements with the same ID. But it’s incorrect, so don’t do it. (getElementByID will, by the way, only return the first element with the specified ID that it finds).

An element can have both an id and a class. It can also have more than one class. For example:

```
<div id="basic_info" class="sidebar pull-right">
```

HTML FORMS:

Forms are used for getting input!

```
<form>
    ...various input elements...
</form>
```

Here are the various input elements you can use:

Button

Just a button. The **`value`** attribute specifies the text to be displayed on the button.

```
<input type="button" value="submit" />
```

Text Field

Used simply for inputting single lines of text.

```
<input type="text" name="zipcode" />
```

Password

A text field with masked characters (asterisks).

```
<input type="password" name="pwd" />
```

Text Area

For inputting multiple lines of text.

```
<textarea rows="10" cols="30"></textarea>
```

Label

For labeling form elements.

```
<label for="username">User Name: </label>
<input type="text" name="username" id=”username” />
```

Radio Buttons

For selecting one choice out of a limited number of choices.

```
<input type="radio" name="sex" value="male" /> Male
<input type="radio" name="sex" value="female" /> Female
```

Dropdowns

Exactly what it sounds like.

```
<select name="preferred_title">
    <option value="">None</option>
    <option value="Mrs.">Mrs.</option>
    <option value="Ms.">Ms.</option>
    <option value="Miss">Miss</option>
    <option value="Mr.">Mr.</option>
    <option value="Dr.">Dr.</option>
</select>
```

Checkboxes

For selecting zero or more options out of a limited number of choices.

```
<input type="checkbox" name="vehicle" value="Bike" /> I have a bike
<input type="checkbox" name="vehicle" value="Car" /> I have a car
```

Input elements should always contain a `name` attribute. This serves two main purposes:

- **Identifying data when it is submitted to the server** The name is submitted with the form data, and essentially becomes the variable name for that input on the server.
- **Specifying that certain form elements are related** For example, the name attribute is used on radio buttons to specify which group they belong to (see the radio button example above). Here, the browser will only allow one radio button from each "group" to be selected at any given time.

HTML FORM EXAMPLE + VALIDATION

Here's a quick example of an html form, with some rudimentary validation built-in thanks to javascript! Try it out yourself here.

```html
<html>
  <head>
    <script type="text/javascript">
      function submitTheForm() {
        var input = document.getElementById('textfield');
        //parseInt returns NaN if it fails to parse a number
        if (isNaN(parseInt(input.value, 10)) === true)  {
          // use regular expressions to do validation better
          alert("Not valid!! You didn't type a number");
        } else {
          alert("Valid! You typed a number");
        }
      }
    </script>
  </head>
  <body>
    <form name="myform" action="javascript:submitTheForm()">
      <input id="textfield" name="Text" type="text" placeholder="Input only a number">
      <input name="Submit"  type="submit" value="Update"/>
    </form>
  </body>
</html>
```

HTML MARKUP VALIDATION

One good way to make sure your HTML is valid is to---validate it! This can be done using the the W3 Markup Validation Service. It's really picky, but it has good advice--listen to it!

WHAT IS HTML5?

It's a **standard**, defined by the W3C. That's it. It describes what HTML *should* do. It's up to individual browsers to implement the W3 spec, and they're not really required to do so. Some browsers have supported most of HTML5 since 2007. Some still don't. HTML5, as of August 2012, was *technically* still under development.

HTML5 NEW FEATURES

HTML5 added a few new tags, simpler syntax for some elements, and a new set of APIs for use with Javascript. Canvas is one of these new APIs by the way! HTML5 was also built with cross-platform applications in mind.

We'll cover many more HTML5 features later in the semester. For now-here's a sample--the new audio and video tags:

`<video>`

Made to replace flash videos, allows video to be directly embedded in websites. Multiple filetypes should be used for cross-browser compatibility. For example:

```
<video width="640" height="360">
    <source src="myvideo.m4v type="video/mp4"/> <!-- Safari MPEG4 -->
    <source src="myvideo.ogg type="video/ogg"/> <!-- Firefox Ogg -->
</video>

```

`<audio>`

Play audio in the browser (great for games). As with **`<video>`**, multiple filetypes should be used for cross-browser compatibility. For example:

```
<audio>
    <source src="song.ogg" type="audio/ogg" />
    <source src="song.mp3" type="audio/mpeg" />
</audio>

```

HTML ESCAPE SEQUENCES

What happens if you want your page to display a character that's usually used in HTML syntax? Like in many other programming languages, you have to use an escape sequence. For example, to display a **`">"`** you would type **`"&gt;"`**. For an exhaustive list of all available characters and their corresponding escape sequences, see this page (look under the HTML Name column).

THE DOM (DOCUMENT OBJECT MODEL

Web Developers like to throw around the word "DOM" a lot. Technically, the DOM is an API (Application Programming Interface) for HTML and XML. It allows you to traverse and interact with it's content through various scripting languages (in our case, Javascript).