**HTML and CSS General Purpose Notes**

**General/Useful Information & General Definitions**

* + Every page and every image on a website has **URL** (Uniform Resource Locator).
  + Save images to be the same size that you intend to have them rendered on your web page. There is probably no need to send a huge file over the internet if you are just going to have the browser resize it down to a smaller image.
  + Embedding – Refers to the integration of links, images, videos, gifs, and other content into web media

**HTML**

**HTML Definitions**

* + Element - Comprises the opening tag and the closing tag and any content in between them
  + Empty Elements - Do not have any words between opening and closing tags.
  + Attributes -- Provide additional information about the contents of an element. They appear on the opening tag of the element and are made up of two parts: a name and a value, separated by an equals sign.
    - Global Attributes – Can be used on any attribute on the page
  + Block Elements – Elements that always appear to start on a newline
    - <h1>, <p>, <ul>, <li>
  + Inline Elements – Elements that appear to continue the same line as their neighboring elements.
    - <a>, <b>, <em>, <img>

**Attributes**

* + id attribute is used to uniquely identify an html element from other html elements on the page. The value should start with a letter or an underscore (not a number or another character)
    - The same ID can’t be used for multiple elements on the same page or else it will not be unique
    - Giving an element a unique ID on the page will allow you to specifically style it with CSS

**Elements**

* + <html> - Indicates that anything between it and a closing </html> tag is HTML code.
  + <body> - Indicates that anything between it and the closing </body> tag should be shown inside the main browser window
  + <h1> - Any words between this and the closing </h1> tag are a main heading
  + <p> -- A paragraph of text appears between <p> and </p> tags
  + <h2> -- Words between this and </h2> indicate another subheading.
  + **Many other exist, but I am too lazy to type them all on here, and I am sure there are more useful guides on the internet**
  + The <a> element creates a link. Users can click on anything between the opening <a> trag and the closing </a> tag. You specify which page you want to link using the href attribute.
  + <br /> is the same as a /n in Python, C, etc. R
  + DOCTYPE tells the browser which version of HTML to use when reading your code. In order to signal that you’re using HTML 5, you use the **<!DOCTYPE html>** tag
  + <div> allows you to group a set of elements together in one-block-level box. It’s kind of liking a defining a function in a programming language, but you don’t reuse the code. It blocks that section of code off and makes it possible to do macro adjustments to the entire block using CSS.
    - You might use a <div> element to contain all the elements for the header of a website (the div element would contain the logo and the navigation)
    - <span> is similar to the div element in that it blocks a section of code off that can be assigned an id or class attribute. It’s typically used with elements where div is not intuitively implemented such as inline text so that CSS styles can be applied to it.
  + <iFrame> or “inline Frame” is an html page either located on the same server or different server or anywhere else on the web. You can use it to render open-source web software or other applications.
    - Has three required attributes, which are src (the URL source of the page you’re referencing), height, and width.
  + <meta> element contains information about the web page. You can provide a description, keywords, whether the page should be shown on search engines, you can include an author, whether the browser should cache the page and for how long it should cache if this option is selected.
    - <description>
    - <keywords>
    - <robots> tells search engine to add this to their search results or not
    - <author>
    - <pragma> tells the browser whether it should cache the page or not and <expires> tells the browser how long to cache the page, if selected.
  + <video> element embeds a media player which supports video playback for the document. You are able to customize the video element in depth by creating your own playback controls, provide different versions of video for browsers that have different sized screens, and tell different parts of a page to change when the video reaches a certain point.
    - src specifies the path to the video
    - poster allows you to specify an image to show while the video is downloading or until user tells browser to play the video
    - preload tells the browser what to do with the video when it loads the page.
      * “none” – browser should not load the video until user presses play
      * “auto” – browser should download the video when the page loads
      * “metadata” – browser should just collect information such as the size, first frame, track list, and duration
    - controls – if supplied, browser will use its own controls for playback
    - autoplay – specifies that the file should play automatically
    - loop – indicates that the video should start playing once it has ended
  + <source> allows you to specify multiple alternative video formats to the browser, in the instance that the only one you supplied/may have supplied would not be supported by that specific browser.
    - type will tell the browser the format of the video
  + <audio> element allows you to include audio on your web page. You’re also able to cite multiple audio like you are for video files.
    - src specifies path to audio file
    - controls tells the browser whether to include playback/volume controls on the web page
    - autoplay
    - preload tells browser what to do if autoplay is not specified
    - loop
  + <link> Can be used in an html document to tell the browser where to find the CSS file used to style the page. It’s an empty element and doesn’t need a closing tag.
    - href specifies the path to the css file
    - type specifies the of document being linked
    - rel describes the relationship between the HTML page and the file that’s being linked

**Lists**

* + Ordered lists -- Lists where each item in the list is numbered. For example, the list might be a set of steps for a recipe that must be performed in order, or a legal contract where each point needs to be identified by a section number.
  + Unordered Lists -- Lists that begin with a bullet point (rather than characters that indicate order)
  + Definition Lists - Lists that are made up of a set of terms along with the definitions for each of those terms.

**Quick and Easy Definitions**

1. Ordered Lists use numbers

2. Unordered lists use bullets

3. Definition lists are used to define terminology

**Links**

Relative Link Type (Very similar, if not identical to a normal filing tree)

* + Same Fodler
    - * To link to a file in the same folder, just use the file name.
      * To link to music reviews from the music homepage use <a href="reviews.html">Reviews</a>
  + Child Folder
    - * For a child folder, use the name of the child folder, followed by a forward slash, then the file name.
      * <a href="music/listings.html">Listings</a>
  + GrandChild Folder
    - * Use the name of the child folder, followed by a forward slash, then the name of the granchild folder, followed by another forward slash, then the file name.
      * To link to DVD reviews from the homepage: <a href="movies/dvd/reviews.html">Reviews</a>
  + Parent Folder
    - * Use ../ to indicate the folder above the current one, then follow it with the file name.
      * To link to the homepage from the music reviews: <a href="../index.html">Home</a>
  + Grandparent Folder
    - * Repeat the ../ to indicate that you want to up two folders (rather than one), then follow it with the file name.
      * To link to the homepage from the DVD reviews: <a href="../../index.html">Homepage</a>

**Tables**

Table Elements

* <table> is used to create a table. The contents of the table are written out row by row
* <tr> indicates the start of each row. It’s followed by one or more td elements (one for each cell in that row) before being closed out with </tr> tag
* <td> represents a cell in the table
* <th> is also a cell entry in the table, but it’s used to denote that it’s a heading for a column or a row

Long Table Elements

Long tables will contain head row, then the body of rows, and then the footer row (or the bottom row) which might contain a total of some kind

* <thead> are where the elements from the head row will be placed
* <tbody> contains the entire body of the table (obv)
* <tfoot> is where the footer element belongs

**Forms**

Browsers sent information to servers using name/value pairs.

Elements

* <form> Denotes that you are starting a form “thread” or whatever the technical term is for when you start creating a new object in a line of html code
* <action> Every form element requires an <action> attribute. It’s value is the URL for the page on the server that will receive the information in the form when it is submitted.
* <method> Forms can be sent using one of two methods: get or post
  + The get method means that the values from the form are added to the end of the URL specified in the action attribute. Ideal for short forms such as search boxes or when you are retrieving data from a web server (not sending any information)
  + The post method means that the values are in what are known as HTTP headers. You should use the post method if the form allows users to upload a file, is very long??, contains sensitive data, adds information to, or deletes information from a database
  + If no method is used, then the form data will be sent using the get method
* <id> identifies the form distinctly from other elements on the page
* <input> creates controls as to what information the user will be sending via the web
  + The type attribute determines how the user will input data (via text, via multiple choice, etc)
  + The name attribute distinguishes what field each packet of information sent by the user belongs to. For example, name = username means that the user is sending their username, and name = password means that the value sent by the user for this form should be regarded as their password
  + Maxlength can be used to limit the number of characters that users can enter into a field
* <textarea> is used to create a multi-line text input… (if you had an “other” response tab, then you probably would use this). Any text between the opening and closing textarea tags will appear in the box when the page loads. If you want it to disappear when the user clicks on it, then you will have to write some JavaScript to take care of that
  + Should use CSS to control the height and width of the textarea box instead of the cols and rows attributes… The rows attribute indicates how many rows the text area should take up vertically… The rows attribute indicates how many cols should take up vertically.
* Input type=”radio” allows users to pick one of a number options. The name value pair is sent to the server when the user selects the option. The name should be the same for all the options and value should match that option the user could potentially pick
  + Checked will choose which radio option is checked when the page is loaded. Can only be used once. If two radio options contain the checked box, then last one to contain it will be the only one recognized by the browser
* Input type = “checkbox” is just a list of options where the users can select multiple or none of the options present. Name:Value pair sent to the server when the form is complete.
  + Name attributes should all be the same
  + Value should be the same as the one checked in the box
* Drop down list box utilizes the <select> struct, which is a drop down list box that allows the users to select one option from a drop down list menu. It will contain two or more elements.
  + The <option> element specifies which options the user can choose from. Words between the <option> and </option> tag are what the user will see when the click on the drop-down menu
  + Selected is the same as checked – it will be the first thing checked if the user does not select anything.
  + **Radio Buttons vs Drop Down Box?** – If users need to see all options briefly, then radio buttons are better. If the list of options are long, then it’s better to use the drop down list box
* When attempting to do a file input, you must have the input type set to file in the input element as well as the method attribute set to “post”.
* The submit button allows you to send a form to the server. The first input statement describes the data that will be submitted, and the second input statement describes the type of input.
  + The value argument will be the text that shows up inside of the button.
* Use the button element to combine text and images between <button></button>
* <label> element can be used to wrap around the text description and the form input. It can
* <fieldset> allows you to group related form controls together, which can be helpful when you’re dealing with longer forms.
  + <legend> comes after the <fieldset> element and is used to identify the purpose of the group of form controls
* Type=”url”, Type=”date”, and type=”email” can be used to check and make sure submissions match any one of these following types. Type=”search” can be used for search queries. You’ll use submit as the value pair associated with all of these. For example:
  + <input type=”url” name=”website”/> and <input type=”submit” value=”Submit”/>
  + <placeholder> can be used on any text input as semi-transparent text that is in the box before the user clicks on it. For example “Click her to search” or something of that nature

**CSS**

**General Information about CSS**

* CSS rule has two parts: A selector and a declaration. The example below says that all p elements should be shown with the Arial typeface.

Declarations have two parts: property and a value

p {font-family : Arial;}

**Selector** **Property Value**

h1, h2, h3 { fontfamily: Arial ;

color: yellow; }

* You can format CSS externally in another file or in an html document using the <style> element
* The reason as to why you would want to use an external document to reference CSS rules is so that you can define all CSS-rules once, and then can reuse the rules across multiple pages. Also makes editing rules across all pages easier.

**Selector RulesTable

Description automatically generated**