HTML

starting a .html file:
<!DOCTYPE HTML>
<head></head>
<body></body>
</html>

tags:

1. tag ( )
- displays content of a paragraph WITHOUT extra white space
- add
 tag for line breaks
- add <pre></pre> to preserve the white space
- whenever a new is started, the browser breaks the current line and inserts a blank line.

2. heading tags <h1>, <h2>, <h3>, <h4>, <h5>, <h6>
- headings use boldface font
- always break current line, so their content appears on a new line
<h1> - highest level (biggest)
<h6> - lowest level (smallest)
- <h1>, <h2>, <h3> use font size generally larger than default size
- <h4> uses default size
- <h5> and <h6> use font size generally smaller than default size

3. <blockquote>, has start tag AND end tag
- made when we need a block of text to be set off from the normal flow of text, aka made to look different from surrounding text
- usually indented (p-inds)
- also makes it easier to manipulate the section using css
- usually a quote

4. font styles and sizes tags, start tags AND end tags
- can be done through html, but easier with css
- tags such as <i> for italics, <b> for bold, and for emphasis
- <code> tag specifies monospace font
- subscript <sub> and superscript <sup>
- these tags are NOT affected by <blockquote>, UNLESS there is a conflict

5. img tags. only start tag
- inline tag
- in its simplest form, includes 2 attr: src (where image is located) and alt (text in case img cannot be displayed)
- but we can also include width and height attrs.
ex: <img src = "images/img.jpeg" alt = "An image">

6. link tags, start and end tags
- inline tag
- links are a specified attr of the anchor tag <a>
- in its simplest form, includes 1 attr: href
ex: clickable area
obs: whatever comes in between the start tag and the end tag is the clickable area (it can be a a text, or a word or an image) > to click on an image:
 <img src = “images/img.png”>
- we can also use links to change the position of the page (make a specific element be the first one in the page)
ex: <h2 id = “avionics”></h2>
text, when the word “text” is clicked, the document will display h2 with id “avionics” at the top of the page

7. lists, ordered, unordered and definition lists (terms and definitions, like glossaries)
- unordered lists: tags, bullet points
- ordered lists: tags, arabic numerals sequential values
- each item in a list is specified with a
obs: if we want to have nested lists, we CANNOT simply add another <ol> or <ul> tag inside another list tag, we need to make it a list item, aka put it inside a <li> element

8. tables, <table></table> tags
- border attr: when any nonzero value is specified for the border, the rules are automatically assigned a value of 1px
- table may have a name, specified with <caption></caption> tags
- each row in a table is specified with the <tr></tr> tags
- each row may have a heading, specified with the <th></th> tags
- each column in a table is specified with the <td></td> tags
-align attr: center, right and left
- for manipulating multiple levels (menu lists), use rowspan and colspan attrs
- colspan: make cell as wide as the specified number of rows below it in the table
Ex:

```
<table border = "border"> <caption> List </caption>
            <tr>
                <th>Fruit Juice</th>
                    <td> Orange </td>
                    <td> Mango </td>
```

```
                    <td colspan = "2"> Guava </td>
            </tr>
            <tr>
                <th>Milkshakes</th>
                    <td> UHHHHH </td>
                    <td>  Strawberry </td>
                    <td> Vanilla </td>
                    <td> Caramel </td>
            </tr>
        </table>
```

List

Fruit Juice	Orange	Mango	Guava	
Milkshakes	UHHHHH	Strawberry	Vanilla	Caramel

-rowspan: same thing as colspan but for rows

```
<table border = "border"> <caption> List </caption>
            <tr>
                <th>Fruit Juice</th>
                    <td> Orange </td>
                    <td> Mango </td>
                    <td colspan = "2"> Guava </td>
            </tr>
            <tr >
                <th>Milkshakes</th>
                    <td> UHHHHH </td>
                    <td rowspan = "2">  Strawberry </td>
                    <td rowspan = "2"> Vanilla </td>
                    <td rowspan = "2"> Caramel </td>
            </tr>
            <tr>
                <th>Milkshakes</th>
                    <td> UHHHHH </td>
            </tr>
        </table>
```

List

Fruit Juice	Orange	Mango	Guava	
Milkshakes	UHHHHH	Strawberry	Vanilla	Caramel
Milkshakes	UHHHHH			

9. forms, <form></form> tags
- required attr: action, specifies url to be called when button is clicked
- get method: attaches query string to the url, limited length
- post method: query string passed through a different method (it doesn't show on the url = more secure)

10. form controls
- <input></input>, attr. type: text, passwords, checkboxes, radio, reset (controls back to initial state), submit (sends info over to server)
- <label></label>: it's just a label
- <select></select>: specifies a menu, if multiple choices, too long to display with input
- <option></option>: specifies each item in drop down menu form similar relationship to select as <li> has to <ol> and
- <textarea></textarea> multiline text area duh, include rows and cols attrs to specify the area

Ex:

```
<form action = "GET">
            <label> Username:
                <input type = "text" name = "username"> </input> <br>
            </label> <br>

            <label> Password:</label>
            <input type = "password" name = "password"></input>
            <br><br>

            <select name = "title">
                <option> Mr. </option>
                <option> Ms. </option>
                <option> Mrs.</option>
            </select> <br> <br>

            <label>Enter some pets here:</label> <br>
            <input type = "checkbox" name = "pets">Dog</input>
            <input type = "checkbox" name = "pets">Cat</input>
            <input type = "checkbox" name = "pets">Bird</input> <br> <br>
```

```
            <label>Gender</label><br>
            <input type = "radio" name = "gender">Female</input> <br>
            <input type = "radio" name = "gender">Male</input> <br><br>

            <label> Please enter some stuff here so I can complete my lab,
thanks</label> <br>
            <textarea name = "client info" rows = "20" cols =
"50"></textarea> <br>

            <button type = "submit" value = "submit">Submit</button>
        </form>
```

forms:
- most common way for a user to communicate info from a web browser to the server

tables:
border: lines around the whole table
rules: lines that separate the cells from each other

images:
- most browsers can support images represented in gif, jpeg and png formats
- gif -> 8 bits, supports transparency but only 256 colours compared to the 16 million for jpegs
- jpeg -> 24 bits, better compression algorithm (smaller) but loses colour accuracy
- png came later and groups the best characteristics of both formats ^ BUT bigger (require more space)

links:
- pointer to some particular place in some web resource
- logically links different resources and pages

character entities:
- special characters that are sometimes needed but cannot be typed (<, >, &)
- & -> &
- < is <
- > is >
- fractions can be represented using &fract(numeratordenominator)
1/4 -> &fract14;
- degree symbol -> °
- copyright symbol -> ©
CSS

types of style sheets: inline (by using style attr per element), internal (body <style>), external (separate css sheets using classes and ids and elements)

Inline, using style property tag:
<article style = “border: solid red 1px;”>

Internal, using <style></style> tags
```
<style>
   p.quote {
      font-family: “Times New Roman”, Times, serif;
      text-align: center;
      font-style: italic;
   }
</style>
```

External, using css file
- same way as internal except no need to define a <style> tag since it’s not html
- need to link the css file to the html using <link> tag, START TAG ONLY
<link rel = “stylesheet” type = “text/css” href = “https:/www.whatever.com/cssfile.css”>

```
<link type = "image/x-icon" rel = "icon" href = "images/icon.png">
<link type = "text/css" rel = "stylesheet" href = "css/core.css">
<link type = "text/css" rel = "stylesheet" href = "css/about.css">
```

Simple selector forms:
- if html element, the css would apply to all elements that match the specified one,
ex: p {} whatever is inside the curly braces apply to all elements in the document

Generic & Class selectors:
- allows us to target groups of elements, less repetition
- if html element grouped by class, the css would apply to all elements that belong to the same class (the elements don’t necessarily need be the same),
<h1 class = “courses”></h1>,
ex: .courses{} applies to ALL html elements belong to class courses - generic
p.courses{} applies to all elements belonging to class courses - class-based

Id selectors:
- allows us to target specific elements
- applies to one specific element (since id cannot be duplicated),
<h1 id = “title”></h1>
ex: #title{}, css inside curly braces applied only to that specific element

Universal selectors:

- applies to ALL elements in a document
- useless piece of shit
* {} css in the curly braces applies to EVERYONE

Pseudo classes:
- styles that apply when something happens rather than because an element is there
ex: hover (applies when mouse hovers over the element) and focus (when mouse hovers and left clicks),
input: hover{}, css inside curly braces is applied when mouse hovers over an <input> element
Input: focus{}, same as ^ but focus
Input [type = submit] {} = styling buttons, when type is “submit”

Font Families:
- used to specify a list of font names
- browser uses the first font in the list that it supports
- use a generic name as last in the list so that if none are supported by the browser, the generic one will still run, ex: sans serif or serif or cursive.
Font-family: “Times New Roman”, Times, sans-serif

Font Sizes
- can range from xx-small to xx-large or be specified in pt
font-size: xx-small;
font-size: 12pt;

Font Styles
- italics, bold, oblique or normal, bolder, lighter
font-style: bold;
font-style: italic;

Font shorthands
- possible to state multiple font properties in one line
font: bold 14 pt “Times New Roman” Palatino

Text decoration
- special features of text: line-through, underline, overline
text-decoration: underline;

Colours:
- 17 named ones, rest use hex or rgb

Colour Properties:
- background colour of html element (refer to automatic box model)
background-color: red;
background-color: #00FFFF;

Text Alignment:
indent, align
text-indent: 2px;
text-align: center;

Borders:
border-style: dashed; (or dotted;)

Margins and Padding:
margin: 5px 5px 5px 4px;
Or
margin-top: 5px;
margin-left: 10px;
Same thing for padding

JAVASCRIPT

General information:
- developed by Netscape in 1995
- capabilities in both client and server side
- to add a script link to an html doc, do
<script type = “text/javascript” src = “game.js”></script>, usually defined in <head>
- pay attention to load time (where methods are called in the html file as well as script being executed)

Javascript objects are dynamic, meaning they can change at run time

Primitive data types:
- number, string, boolean, undefined and null.
- wrapper classes for these are also a thing
- exponents with lowercase or uppercase e, 3e4

Variables
- JS is dynamically types so a variable can have its type change at any point. Used for anything.