**HTML file**

<!DOCTYPE html>

<html>

<head>

<title>My first blog post</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<h1 align="center">HTML and CSS</h1> <br>

<h2>HTML Outline</h2>

<p>

</p>

<p>

HTML Documents are described by HTML Tags <br>

To write in bold, use the <strong>"strong"</strong> tag. <br>

Use <strong>"em"</strong> (emphasize) tag for <em>Italics / emphasize</em> <br>

use <strong>"u"</strong> tag for <u>underlined</u> text. <br>

<br>

Tags mark up the beginning and end of an element, using the general syntax: <strong>"tagname" content "/tagname"</strong> <br> Where the tagnames are enclosed in less-than and greater-than brackets and not the quotes used for example. <br>

</p>

<h3> Basic HTML Structure</h3>

<p>

The first thing in the HTML file should be <strong>"!DOCTYPE html"</strong> which tells the browser the document's type is html5. <br>

then <strong>"html"</strong> tag <br>

Indent heading and use the<strong>"head"</strong> tag, which contains the tags and elements: <em>title, scripts, (css) styles, and meta info</em> - search engine info. <br>

Keeping same indentation, use the <strong>"body"</strong> tag for anything you want to be directly visible on the web page. <br>

Indent again (2 indents) for <strong>"h1" "h2" ... "h6"</strong>(h6 is smallest) to write a Header Tag.<br>

Use Paragraph <strong>"p"</strong> tag for text body like this paragraph.

</p>

<img src="logo.jpg" alt="HTML Syntax and Structure">

<p>HTML Syntax above</p> <br>

<h6 align="right">This is a "h6 align="right" " header</h6>

<img src="tag\_Attributes.jpg" alt="Opening Tag Attributes">

<p>Attributes are just additional info about an element.</p>

<h4>To insert an image use the <u>"img" tag with attribute 'src'</u></h4>

<img src="html-attributes.jpg" alt="HTML Attributes"> <br> <br> <br>

<h3>To insert a hyperlink use <u>"a" tag with "href=" </u></h3>

<p>

For example: <strong>"a href="http://www.SiteName.com">Hyperlink display name" "/a"</strong> <br><br>

To have link open in a new tab use <strong>targets="\_blank"</strong> after the <em>href=</em> link. Which can be used for a website or even an image. <br>For example: <br><br>

<strong>"a href="logo.jpg" target="\_blank">HTML Syntax "/a"</strong>

</p> <br>

<a href="https://www.w3schools.com/tags/" target="\_blank">HTML Tags Reference List</a>

<h2>Cascading Style Sheets (CSS)</h2>

<p>

CSS defines exactly how HTML looks and allows to make a clear separation between <strong>content (HTML)</strong> and <strong>style (CSS)</strong> <br> <br>

<strong> Three ways to use CSS</strong> <br> <br>

1. CSS code written insdie an HTML tag using 'style='. (i.e., p style="font\_size: 120%") <br> <br>

2. CSS code written in the head section of HTML doc by using the <strong>"style"</strong> tag. <br> (i.e., "style" <br> (indent) p {<br>(indent)font-size: 120% <br> (dedent)} <br> (dedent) "/style") <br> <br>

3. <strong>Most Practical -</strong> CSS can be put into an external file. (i.e., open new file and save it as "style.css") <br><br>

</p>

<h3>Linking style.css to your index.html file</h3>

<p>

In the <strong>"head"</strong> tag section of your html file, use the <strong>"link"</strong> tag, which doesn't have an ending tag ("/link" is not used) <br><br> For Example: <br> <br>

link rel="stylesheet" type="text/css" href="style.css"<br><br>

</p>

<h3> Coding inside the CSS file itself</h3>

<p>

CSS itself is written in rules, which consists of a <strong>selector</strong> and a <strong>declaration block</strong> <br><br>Start by formatting the main heading, using the selector "h1" which will select all "h1" elements and format them. <br>For Example <br> <br>

h1, h2 { This is for things h1 and h2 have in common<br>

color: green; <br>

font-family: Helvetica Neue, Arial;<br>

}<br><br>

h1 {<br>

font-size: 40px;<br>

}<br><br>

h2 {<br>

font-size: 25px; <br>

} <br><br>

<strong>Quick Edit</strong> a CSS file while inside your HTML file using <strong>ctrl+E</strong> on the tag you want to format.

</p>

<h3>The RGB Model</h3>

<p>

In the RGB Model, each color is defined using a combination of Red, Green, and Blue. Hexadecimal notation is used #RRGGBB <br><br>

<strong>For each color:</strong><br> Min: 0 (hex 0)<br> Max: 255 (hex ff)<br>

</p>

<img src="hex-colors.jpg" alt="The RGB Model">

<h4>Some Hexadecimal color codes</h4>

<img src="hex-codes.jpg" alt="Hexadecimal Codes"> <br> <br>

<p>If all 6 numbers in hexadecimal code are the same, instead of typing 555555 we can write 555</p>

<h4>Transparency with RGBA</h4>

<p>

Ex: rgba(29, 167, 23, <strong>0.75</strong>) <br><br> This shows a green rectangle with <strong>75% transparency</strong> <br><br>

You can adjust each color in the CSS file by clicking the color name or hex and using the hot key <strong>ctrl+E</strong> which opens up a quick edit for that color. Here you're able to adjust the transparency and color shade.

</p>

</body>

</html>

**CSS file**

body {

font-family: Helvetica Neue, Arial;

font-size: 18px;

}

h1, h2 {

color: #0c840c;

}

h1 {

font-size: 40px;

}

h2 {

font-size: 25px;

}

p {

text-align: justify;

}

**CSS file from instructor of course**

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

.clearfix:after {

content: "";

display: table;

clear: both;

}

body {

font-family: Helvetica Neue, Arial;

font-size: 18px;

}

h1, h2 {

color: #00960b;

}

h1 {

font-size: 40px;

margin-bottom: 20px;

}

h2 {

font-size: 25px;

margin-bottom: 10px;

}

.main-text {

text-align: justify;

margin-bottom: 20px;

}

.author-text {

font-size: 22px;

float: left;

margin-top: 30px;

margin-left: 10px;

}

.container {

width: 1140px;

margin: 20px auto 0 auto;

}

.blog-post {

width: 75%;

float: left;

padding-right: 30px;

position: relative;

}

.other-posts {

width: 25%;

float: left;

}

.author-box {

padding-top: 20px;

border-top: 1px solid #808080;

}

.other {

margin-bottom: 40px;

}

.author-box img {

height: 100px;

width: 100px;

border-radius: 50%;

float: left;

}

.blog-post img {

height: 150px;

width: auto;

}

.date {

position: absolute;

top: 10px;

right: 30px;

}

p {

font-size: 18px;

text-align: justify;

font-family: Helvetica Neue, Arial;

}

**Classes and ID’s - Section 3, Lecture 15**

We use classes or IDs when we want to style only one of the paragraphs, but not all of them.

We can attribute class or ID names to elements and use the classes or IDs to select them in the CSS code.

The same class can be attributed to as many elements as you’d like, while an ID can be used only once inside each HTML document.

The class or ID must be defined in the CSS file and referenced in the HTML file.

For example, **referencing the class or ID in HTML**:

<p class="main-text"> **Class** used for the main paragraph

<p id="html-syntax"> **ID** used for just one single paragraph

The names in quotations are just variables of your choice, or they’re the class/ID variable if you created it in the css file first.

An example of **defining the variable in the CSS file:**

.main-text { **.class\_name** defines the class variable (using the dot “.”)

text-align: justify;

}

#html-syntax { **#ID\_name** defines the ID variable (using the hashtag “#”)

font-size: 22px;

}

Typically, using IDs is not good practice. It’s better to use classes for all styling, even if only using one class with a specific name.

**The CSS Box Model**A screenshot of a cell phone

Description automatically generated

**Box Model Properties in CSS**

* **Content:** text, images, etc…
* **Padding:** transparent area around the content, inside of the box.
* **Border:** goes around the padding and the content
* **Margin:** space between boxes.

What the box model does is allow us to define space between elements and add a border around elements.

With **box-sizing**: border-box in CSS, we can define the **height** and the **width** for an entire box, rather than just for the content.

If we set box-sizing to **border-box**, the height and width will be defined for the entire box.

In HTML, there are **block** elements and **inline** elements:

**Block** elements use the full width of the browser and force line breaks. Headings and paragraphs are examples of block elements.

**Inline** elements don’t do any of the things block elements do. For instance; images, links, strong (bold) and “em”(emphasize/italics) are all inline elements. What is important about them is that you can either set their height OR their width.

All browsers add some default margins. Default values can be removed by creating some rules in the .CSS file that affect all elements on the web page by using the **asterisk** (“\*”).

For example, **removing the margins, padding, and setting box-sizing to border-box** for all elements:

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

And in turn, we must set margins for specific elements that we want to have margins.

For example:

h1 {

font-size: 40px;

margin-bottom: 20px;

}

h2 {

font-size: 25px;

margin-bottom: 10px;

}

.main-text {

text-align: justify;

margin-bottom: 20px;

}

**Using the <div> element in HTML**

The **div** element divides the page into sections by creating boxes where we put our content in.

For example:

<!DOCTYPE html>

<html>

<head>

<title>Title name</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

**<div class=”container”>**

**<div class="blog-post">**

<h1 align="center">HTML and CSS</h1> <br>

<h2>HTML Outline</h2>

<p>

~~~~~~~~~~~~~~paragraph text ~~~~~~~~~~~~~~~~~~~~~~~

</p>

<img src=”………..” alt=”…………”>

<a href=”https://www.site.com” target=”\_blank”> Hyperlink name</a>

**</div>**

**</div>**

</body>

</html>

The **class** named “container” must be defined in the CSS file as follows:

.container {

width: 1140px;

margin: 20px auto 0 auto;

}

This will center the contents of “container” in the window, since we defined the left and right margin of the container to be auto. This **auto** means the left and right margin are adjusted automatically according to the content of the element, which is the browser window in this case.

.other-posts {

background-color: #00ff00;

width: 25%;

float: left;

}

**float** pushes element to ‘left’ or ‘right’ allowing other elements to wrap around it

.author-box {

background-color: #ffff00;

padding-top: 20px;

border-top: 1px solid #808080; **border-top:** width = 1px, line type = solid, color

}

**Padding-top** to add white space inside the box.

Padding is between border and the content.

.container {

width: 1140px;

margin: 20px auto 0 auto;

}

**margin syntax -**  margin: top right bottom left;

So “.container” has margin with top=20 pixels, right = auto fit, no bottom margin, and left is auto fit.

.blog-post {

background-color: #0000ff

width: 75%; **width: 75%;** sets class width to 75% of container’s

float: left;

padding-right: 30px;

position: relative;

}

Since the division with class of “blog-post” is inside the “container” division, the blog-post div will have the same width as the container, but specifying width: 75% will now make blog-post’s width 75% of what the container’s width is.

.clearfix:after {

content: "";

display: table;

clear: both;

}

We use the **clear** property in the “.clearfix” class to clear the float so that the ‘author-box’ can be right under the other

**Element Positioning: Relative vs Absolute**

So far, all elements have been **relative** positioned elements. Meaning that their position on the page is determined by other elements.

Elements with **absolute** position can be positioned anywhere we want inside their parent elements.

For instance, if we wanted to place a date above the blog post. <p class="date">May 14th, 2019</p> at the top of our page, but below the main heading.

**Google Chrome Developer Tools**

open the actual file, right click, open with google chrome, where you can inspect a given element to access the developer tools.