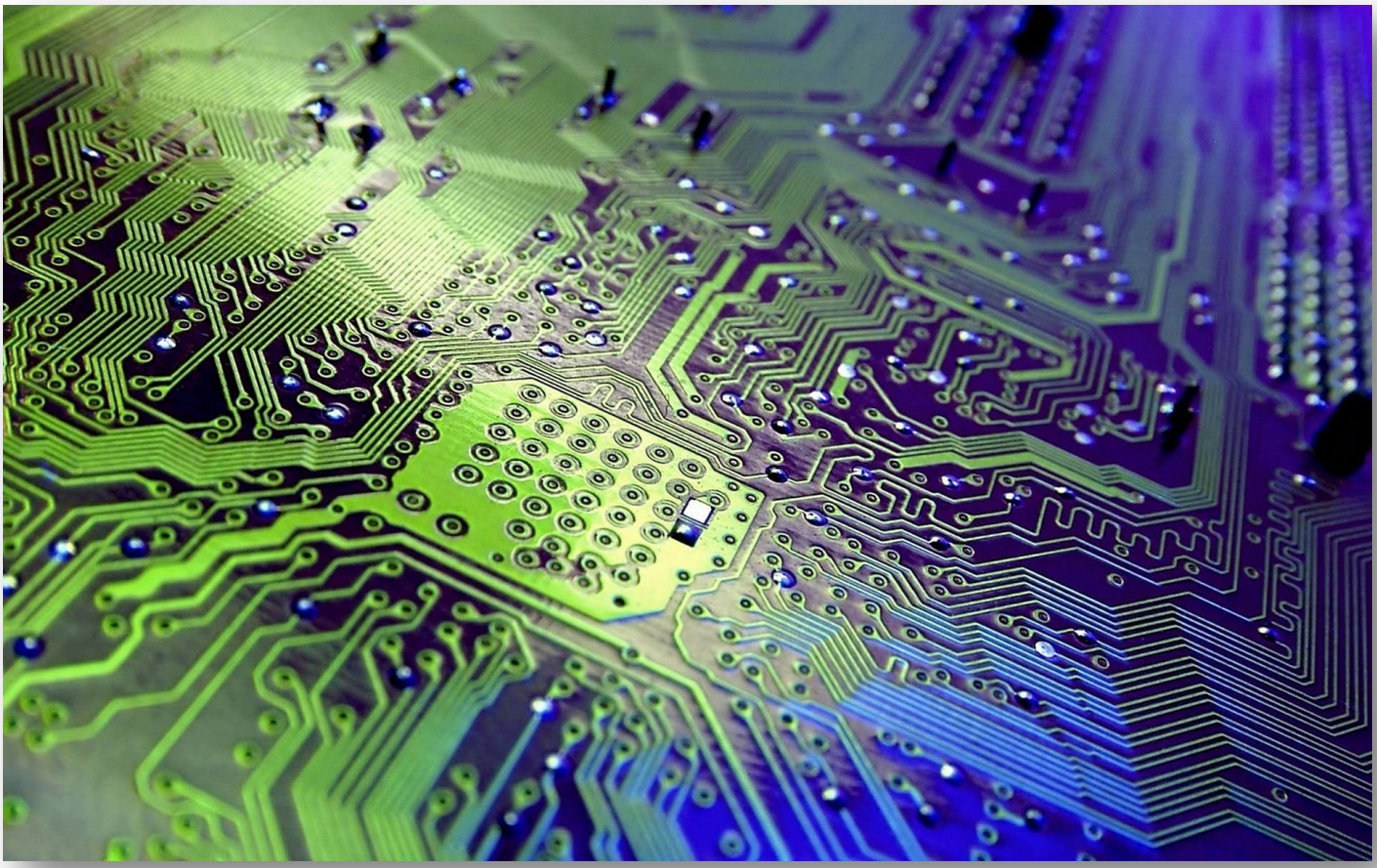
HTML5 and CSS for Beginners



Created by: S.J. Aragon
Created for: ITLS 4320
Project 1 Assignment 4
February 11, 2023

Contents

I. Overview

HTML5 is the 5th version of the website authoring language, Hypertext Markup Language. HTML5 incorporates the functionality of all previous versions of HTML while integrating new technologies such as text-to-speech for the visually impaired, speech-to-text for the hearing impaired, video, 3-D graphics and more. HTML5 defines structure, layout, and functionality using characters contained within tags "< > < />" that have attributes telling browsers how to render the webpage. External CSS is included in this How-to Guide (hereafter referred to as "Guide") in order to separate content from formatting as recommended by the W3C Style Guide.

The World Wide Web Consortium (W3C) is the standards body who decides, publishes, and implements HTML and CSS rules worldwide. Its purpose is to expand website accessibility to more audiences by standardizing web development rules. Adhering to W3C rules increases search engine optimization (SEO) and website visits (theoretically).

Web developers (coders) are the target audience for this Guide. Experience level is beginner. HTML5 is open-source and freely accessible to people willing to learn it regardless of socioeconomic background, culture, language, geographical location, technological skill or physical impairment. It has the potential for expanding peoples' information spheres and economic influence concerning commercial or nonprofit ventures. Other benefits include:

- Autonomy with free HTML hosting domains;
- Output control as far as not having to pay for hosting domain templates that are only free for a limited time (i.e., Wix.com and Godaddy.com);
- Free or low-cost software (although a computer is required).

Besides HTML5 integrating new technology capabilities, the other major difference between it and earlier versions of HTML are the container labels. HTML5 uses <head>, <body>, <header>, <main>, <section>, <article>, <aside>, and <footer> container tags whereas previous HTML versions only used container tags with classes to differentiate styles for various containers. Classes are still required in HTML5. However, older browsers cannot read HTML5 tags and require JavaScript to render. HTML5 renders all HTML container tags. Note that although HTML5 and HTML use different naming conventions for containers, they are all still containers. The terms are used interchangeably in this Guide.

This Guide includes instructions for creating an HTML5 webpage with nested containers and instructions for creating and attaching an external CSS. Classes are discussed in Section II, but not used in the HTML5 code in Section II. (See Appendix B for code with classes.) This Guide uses the actual code for a scary story website published on Github.com for a website design class created by the Guide author. The stories have been omitted for length. The styles used for this website are appropriate for the content, but would not be appropriate for other types of content. Therefore, it is recommended that coders learn basic graphic design principles. (See Appendix B for an alternative design.)

This guide includes two appendices. Appendix A contains HTML and CSS descriptions for the code used in this Guide in Section II followed by a "Resources for Further Development" section. These resources are free, except for certification. Appendix B is an alternative layout with two columns and an internal CSS. This layout is more complex than beginner, but is included to show how classes work in HTML and CSS.

II. Create an HTML5 Webpage

1. Structure, Layout and Content

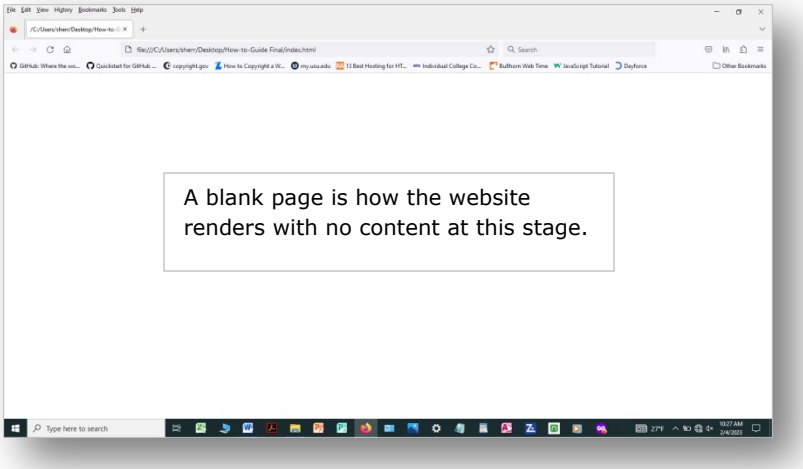
a. Decide structure, layout and content of the webpage.
b. The structure of this site uses nested containers with space-holders for body text and images. Nested containers give the effect of layers. (Containers for this site are different sizes, but all the same color, so layers are not perceivable.)
c. Navigation will be nested in the header.
d. A footer section will be included.

2. Homepage

a. Create the index.html page and save it in a new folder.
b. Open Notepad. Save as: “index.html.” Save as Type: “All Files.”
c. Close the index.html file. Right-click the file. Open with: Notepad.
d. Insert nested containers shown below with these tags in this order exactly. This step creates the space for content. At this stage, the webpage will display as a blank page in a browser (shown below).
e. Notice how the header, main, section, and article containers are nested inside the body container and how the opening and closing tags are ordered inside each element (from the outside in). These are containers nested in their proper order. Any deviation will result in content not being displayed or styled properly.

3. Nested Containers Structure

```
<!DOCTYPE html>
<html lang="en">

<head>
</head>

 <body>
  <header>
   <nav>
   </nav>
  </header>

  <main>
  <section>
  <article>
  </article>
  </section>
  </main>
  <footer>
  </footer>

 </body>
</html>
```



4. Navigation

a. Insert the navigation <nav> </nav> tags inside the <header> </header> container.
b. Coders will often put the <nav> container in a left column. This page uses the header for navigation. (See Appendix A for an alternative format with columns.)

5. Content tags

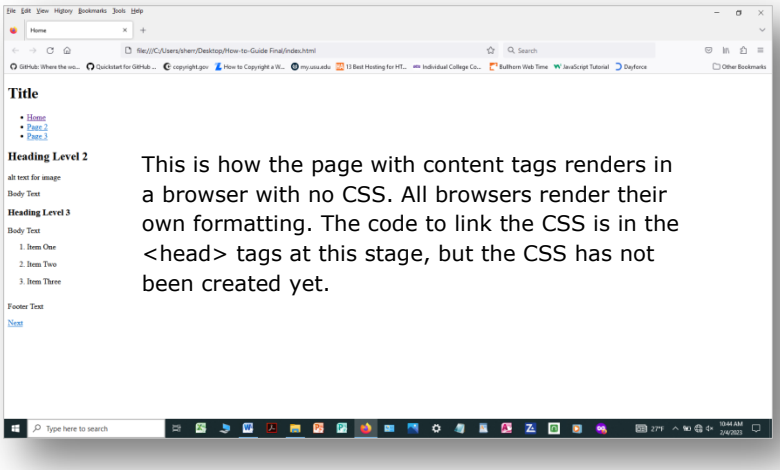
a. Create content tags within the container tags and insert content (example below).
b. Content should be written in a word processing program to spell/grammar check before insertion because Notepad has neither function.
c. Build every section of the Homepage so that all elements of subsequent pages will preexist and not need to be manually recreated on every single page. Subsequent pages will be copies of the Homepage.
d. When naming pages always include the extension (e.g., “page2.html”) or the page will fail to display.
e. Use an opening < >and closing </ > (single angle quotations and forward slash) for container and content tags in HTML so that content renders properly.

6. Spacing

a. Add one to five spaces before every opening tag in HTML to indent sections in order to make it easier to find specific elements when editing code (optional).
b. Some coders use tabs to indent. While W3C recommends not using tabs, there is no standardization for HTML spacing. Hard returns and spaces to separate elements do not render in HTML. Spacing is coder preference.
c. Lines of code should be short so that coders do not have to scroll to the right to edit code. Coders can break up code anywhere they choose and wrap it to the next line.

7. Example of Nested Containers with Content Tags

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport"
content="width=device-
width, initial-scale=1.0">
   <link rel="stylesheet"
href="styles.css">
    <title>Home</title>
</head>

  <body>
  <header>
   <h1>Title</h1>
    <nav>
     <ul>
      <li><a
href="index.html"
```



This is how the page with content tags renders in a browser with no CSS. All browsers render their own formatting. The code to link the CSS is in the <head> tags at this stage, but the CSS has not been created yet.

```
target="_blank">Home</a></li>
     <li><a href="page2.html" target="_blank">Page 2</a></li>
     <li><a href="page3.html" target="_blank">Page 3</a></li>
   </ul>
   </nav>
 </header>

<main>
 <section>
  <article>
  <h2>Heading Level 2</h2>
<div><img src="filename.jpg"
   alt="alt text for image"></div><br>
  <p>Body Text</p>
   <br>

  <h3>Heading Level 3</h3>
  <p>Body Text</p>
   <ol>
    <li>Item One</li><br>
    <li>Item Two</li><br>
    <li>Item Three</li><br>
   </ol>

 </article>
 </section>
  </main>

<footer>
 <p>Footer Text</p>
  <a href="page2.html">Next</a>
</footer>

</body>
</html>
```

8. Create Pages

a. Create subsequent pages (index.html is always the Homepage and tells the browser that the site begins with this page).
b. Copy the index.html file and paste in the same folder.
c. Save as “page2.html.” Repeat for the desired number of pages naming in sequential order.
d. Be sure and use lowercase letters with no spaces when naming files in HTML. Browsers do not recognize spaces in HTML filenames. While browsers do recognize uppercase letters in filenames, Unix and Apache web servers will not display HTML files beginning with uppercase letters.
e. Link subsequent pages in the <header> section inside an unordered list that is nested in the <nav> section as shown on pages 3-4. Everything inside the <nav> </nav> tags is the site’s navigation.

9. Understanding IDs and Classes

Understanding the use of classes and IDs in HTML and CSS is important because CSS applies formatting to all elements with the same tags in HTML uniformly. This means that if coders need to apply different formatting for one container, but not others, they would need to create a class for that container specifically. Use classes for elements in an HTML webpage that appear repeatedly (e.g., lists, containers, font, background colors, etc.). Use IDs for elements in an HTML webpage that appear only once (e.g., the website header). (This webpage does not use classes. See Appendix A to view use of several classes.)

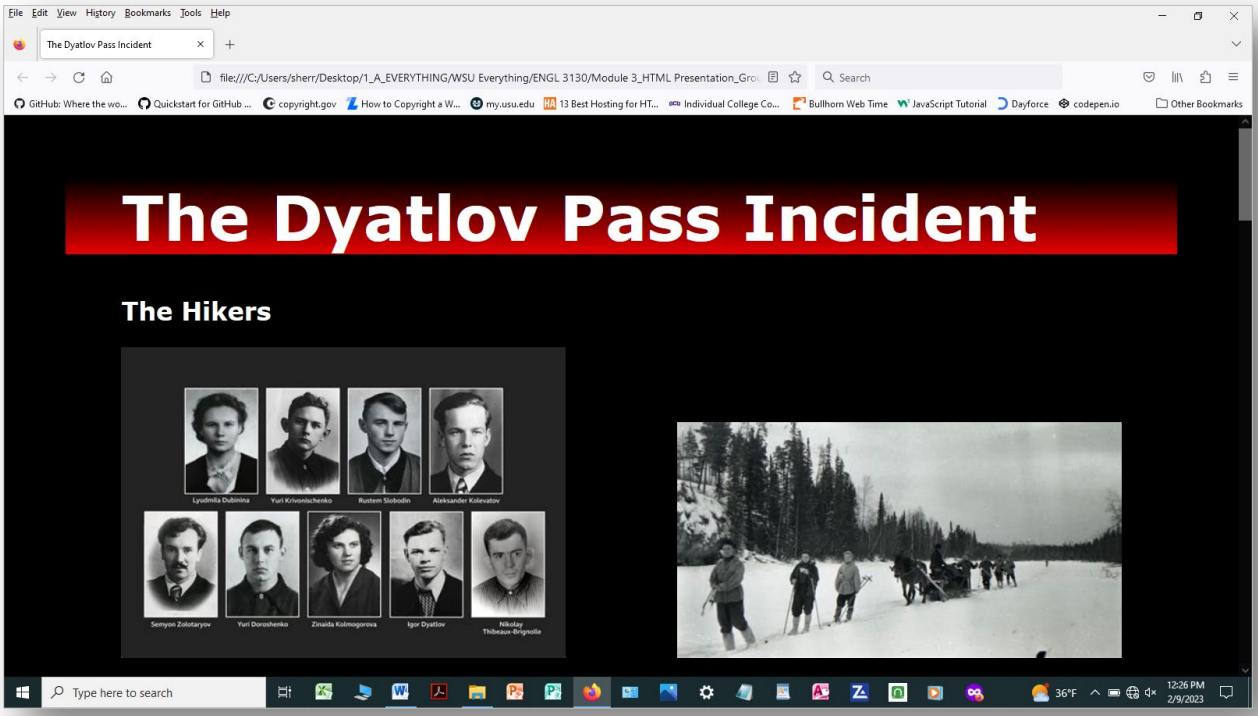
IDs

Below is an example of an ID code entry in HTML. The result will be a red and black gradient colorization in the header container only. There is only one website header.

<header id="grad">

This is an example of the corresponding code entry in CSS and a screenshot of how it renders in a browser:

```
#grad {
background: linear-gradient(to bottom, #000000 0%, #e60000 100%);
}
```



Classes

Classes are used frequently in HTML. Below is an example of a class code entry in HTML. The result will be a bulleted list with square bullets. A separate class needs to be created for this <ul> because the <ul> in the <nav> section of this webpage is an unordered list with no bullets (list-style-type: none;). That formatting would be applied to all other unordered lists in this webpage without a class designation that differentiates this unordered list from the <nav>unordered list.

```
<div class="indent">
 <ul class="square">
  <li>Item 1</li><br>
  <li>Item 2</li><br>
 </ul>
</div>
```

- Item 1
- Item 2

This is how the code on the left renders in HTML.

Here is an example of the corresponding code entry in CSS:

```
.square {
 list-style-type: square;
}
```

For every class and ID designation, there will be two code entries: one in the HTML code and one in the CSS code. (Note: The word after the "#" and "." representing ID and class, respectively, is invented (made-up) by the coder.)

III. Create an External CSS

10. Structure of CSS

a. CSS uses different opening and closing tags than HTML. These { } are curly brackets.
b. Rulesets in CSS do not need to be in any specific order like HTML, but the syntax is important.
c. Once created, the CSS is linked to the HTML doc. It contains the formatting for the website.

11. Create the CSS

a. Open Notepad. Save as: "styles.css." Save as Type: "All Files."
b. As in HTML, do not capitalize or use spaces when naming CSS files.
c. Be sure the CSS is linked in the <head> section of the HTML doc. Use this code: <link rel="stylesheet" href="styles.css"> (See Page 3 for an example.)
d. Below are the CSS rulesets used in this webpage. (See Appendix A for descriptions.)
e. Note: If styles have the same specificity (as in this CSS, "h1" padding has two entries), the latest entry takes precedence. Order of precedence for selectors is: tag, class, id, inline styles. This guide does not use inline styles, but this is the code for a h1 inline style inserted in the HTML code as an example:
<h1 style="color:blue;text-align:center;">This is a heading</h1>.

12. Example of a CSS

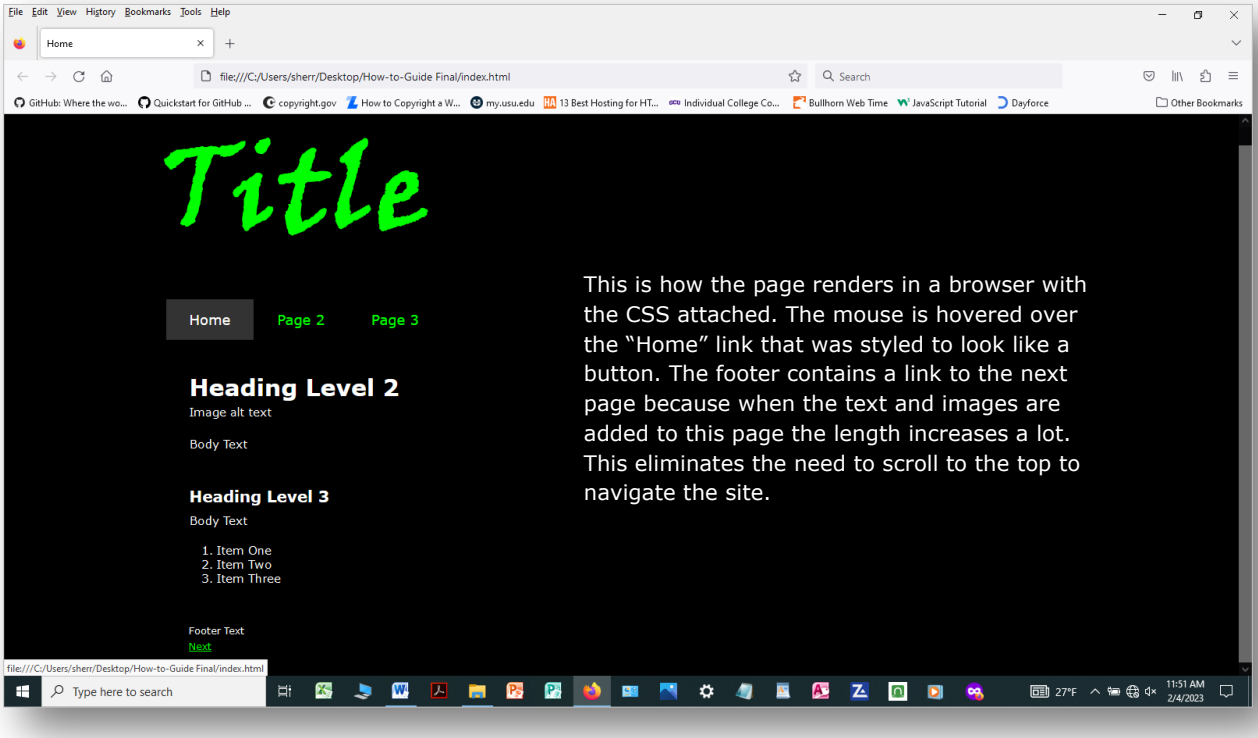
```
@charset "UTF-8";

article, aside, footer, header, main, nav, section {
  display: block;
}

html, body, h1, h2, h3, ul, a, p,
article, aside, footer, header, main, section {
  padding: 5px;
  margin: 0px;
}

h1 {
  font-family: Viner Hand ITC;
  font-size: 148px;
  color: #00ff00;
  padding: 0px;
}

h2 {
  font-size: 30px;
  color: #ffffff;
}

h3 {
  font-size: 20px;
  color: #ffffff;
}

body {
  width: 1200px;
  margin-left: auto;
  margin-right: auto;
  background-color: #000000;
  font-family: Verdana;
  font-size: 15px;
  color: #ffffff;
}

nav {
  background-color: #000000;
  padding: 1px;
  margin-left: auto;
  margin-right: auto;
  overflow: hidden;
  background-color: #000000;
  font-size: 18px;
}

ul {
list-style-type: none;
}

li {
```

```
  float: left;
}

li a {
  display: block;
  color: #00ff00;
  text-align: center;
  padding: 15px 30px;
  text-decoration: none;
}

li a:hover {
  background-color: #333;
}

a:link {
  color: #00ff00;
  margin-left: 0%;
}

a:visited {
  color: #66ff66;
}

a:hover {
  color: #ffffff;
}
a:active {
  color: #80b3ff;
}

* {
  box-sizing: border-box;
}

img {
  width: 50%;
  height: auto%;
  float: left;
  margin-left: 5px;
}

section {
  background-color: #000000;
  margin-top: 10px;
  padding: 1px;
}

article {
  background-color: #000000;
  margin-top: 0px;
  padding: 10px 15px;
}

main {
  width: 900px;
  float: left;
```

```
  margin-bottom: 10px;
  margin-left: 15px;
}

footer {
  clear: both;
  background-color: #000000;
  color: #ffffff;
  font-size: 12px;
  padding: 5px 20px;
  margin-left: 15px;
}
```

13. Test the Site

a. Open the index.html file in a browser to visually inspect layout appeal when rendered.
b. Check alignment of all elements (containers, text, images, etc.)
c. Click all links throughout the site to verify functionality.
d. Make necessary adjustments.



14. Common Mistakes

The most common mistakes coders make are:

- Forgetting to add closing tags to a line of code. This results in code that does not render correctly.
- Forgetting to change the <title>Home</title> tag in the <head> </head> section for subsequent pages. This results in every open tab being titled “Home” when titles of tabs should change on every page according to content. Only one Homepage exists in a website.
- Forgetting to add the file extension (e.g., “.html”) when linking subsequent pages to the website.
- Naming links differently than source file names. If subsequent page links are not working when clicked from the Homepage, the name of the page in the link is inaccurate. All link names must be identical to the source file name. Do not begin the first word of a page name with a capital letter or use spaces in page names. Use dashes or underscores to separate words if desired.
- Deleting an end tag or a line of code accidentally. Use Ctrl+z to Undo and Ctrl+y to Redo. In Notepad, this only works for the last action performed. Be careful.

IV. Conclusion

The HTML5 and CSS code in this Guide actually render if copied and pasted into Notepad and saved as index.html and styles.css, respectively. Open the index.html page in a browser to see how the page renders. Formatting and content should be separated for two reasons:

1. Linking an external CSS means that coders only make one change in the CSS for the changes to automatically cascade throughout all pages of a website after saving.
2. Internal styles can be embedded in the HTML page with <styles> </styles> tags inserted in the <head> section or with inline CSS. The problem with using internal styles is that coders will have to find and edit the same CSS code on every single page of the site manually, which is inefficient.

Once coders have built several websites with different layouts, copying and pasting sections of code from previous projects is more efficient than manually typing code. However, for people who would rather use graphical user interfaces (GUIs), Adobe Dreamweaver, Wix.com, and Godaddy.com use GUI drag-and-drop templates. HTML and CSS are not GUI environments.

All browsers have built-in HTML formatting meaning that HTML webpages will display and function without CSS. However, formatting will display differently in every browser and will not render as nicely as with CSS formatting.

While it is possible to omit all tags except <body> tags and to place all content inside the body element, all content should be placed inside separate containers so that it can be styled and aligned differently without having to use a multitude of classes.

Appendix A

HTML (Hypertext Markup Language) Descriptions	
<!DOCTYPE html>	Tells the browser that this is a webpage.
<html lang="en">	Tells an electronic reader (for the visually impaired) the language of the content.
<head> </head>	Contains instructions for the browser only. This container does not show on the website.
<meta charset="UTF-8">	Allows text to be written in many languages. This attribute specifies the character encoding for the HTML document.
<meta name="viewport" content="width=device-width, initial-scale=1.0">	Tells a browser to scale to the dimensions of the viewport (i.e., cell phone or tablet). Metadata do not have forward slashes, /, in closing tags.
<link rel="stylesheet" href="styles.css">	Tells the browser to link the CSS to this HTML doc. The href attribute specifies the location of any resource the anchor tag points to. The src attribute specifies the location of any resource that needs to be embedded into the present HTML document (including images).
<title>Home</title>	Displays the title in the browser tab of the open window.
<body> </body>	Tells electronic readers where to begin reading on the website for visually impaired users. All body text, navigation, images audio and video reside inside these tags.
<header> </header>	Used for the title of the website and site navigation.
Title	Use tags for the title of the website.
<nav> </nav>	Insert navigation links inside <nav> </nav> tags.
	Insert unordered lists inside tags. HTML navigation is very often just an unordered list with mouse-hover styles applied to render the illusion of buttons. HTML does have button capabilities, but that's advanced (not beginner).
Home	Tells the browser that this is a link to the homepage from another page of the website. This is a list item in the <nav> section.
Heading 2	Precede all topics with heading tags.
Body Text	Insert body text between paragraph tags.
 	Inserts a line break between list items or text. Requires no closing tag.
<footer> </footer>	Inserts a footer section at the bottom of the webpage. Used for links, publication data, author, contact, copyright info, etc.
</html>	Tells the browser where the end of the page is.

CSS (Cascading Style Sheet) Descriptions	

CSS Syntax	Tells the browser which element to style and how.
	
	selector {
	property: value;
	property: value;
	}

CSS Rulesets	Encompasses all instructions inside curly brackets { } for one HTML element.

section {
background-color: #000000;
margin-top: 10px;
padding: 1px;
}
	Tells the browser that everything inside the <section> </section> tags will have a background color of black. The top margin will be 10 pixels. All other margins: margin-bottom, margin-right, margin-left are default. The padding (spacing) of the <section> container will be one pixel wide for padding-left and padding-right.

Main, section and article containers	Styling these containers differently to create layers and inserting content into all three of these containers is possible. These containers are all nested inside the body section and styled the same for this website example for simplicity.

body {
width: 1200px;
length: auto;
background-color: #000000;
font-family: Verdana;
font-size: 15px;
color: #ffffff;
}
	Tells the browser the width of the <body> container is 1200 pixels and the length stops at the end of the last element of content. The background color is black. The font is Verdana. The font size is 15 pixels. The font color is white. (Note that using percentages, flex boxes, and responsive text and images automatically resizes elements according to different viewports and are alternative options to pixels, which are a fixed sizes.)

img {
width: 50%;
height: auto%;
float: left;
margin-left: 5px;
}
	Tells the browser that the width of the image is 50% relative to its container with the height set to auto to protect aspect ratio. The image is left aligned relative to its container with a left margin of 5 pixels. Note that images and PDFs are not search engine readable (use alt text to improve SEO). All images should be placed inside containers and optimized. Optimization improves load time. See https://kinsta.com/blog/optimize-images-for-web/ for more information.

CSS hex number example: #8600b3	Using hex numbers like #8600b3 will render exactly one shade of purple in every browser. Using text in CSS like “color: purple;” instead of “color: #8600b3;” renders a different shade of purple in every browser. See https://www.w3schools.com/colors/colors_picker.asp for the hex number color picker.

ul {
list-style-type: none;
}
	Tells the browser to display no bullets in this unordered list.

CSS Descriptions (Continued)	

li {
  float: left;
}
	Tells the browser to align all list elements to the left of the container.

li a {
  display: block;
  color: #00ff00;
  text-align: center;
  padding: 15px 30px;
  text-decoration: none;
}
 	
li a:hover {
  background-color: #333333;
}
	Tells the browser that for the list of links in the <nav> section to display all list items with a square background when mouse-hovered with text color bright green #00ff00. Text will be center aligned within those squares. Padding around those squares is proportionate within the container so that those squares are distributed horizontally and evenly (can also use positioning and flex box to distribute elements evenly). The text decoration does not include a hyperlink underline (which is the style default for all links).
	
	Hover color is dark gray #333333 to give the <nav> list items the illusion of being buttons.

a:link {
  color: #00ff00;
  margin-left: 0%;
}
	Tells the browser to display the text for all links on the page as a specific shade of green. The left margin is 0% while all other margins are default. If a margin is not mentioned specifically in CSS, it will always be default. Rulesets with colons in the selector are pseudo classes.

a:visited {
  color: #66ff66;
}
	Tells the browser that after a link has been clicked, the text will be another shade of green to show that it has been visited.

a:hover {
  color: #ffffff;
}
	Tells the browser that when a link is hovered over by the mouse, the text will be white.

a:active {
  color: #80b3ff;
}
	Tells the browser that the active link will display a specific shade of green.

* {
  box-sizing: border-box;
}
	Setting border-box on an element means that margin, padding and border are included in the width and height. To calculate total element width = width + left padding + right padding + left border + right border + left margin + right margin. To calculate total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin.

Resources for Further Development

HTML and CSS Code Verifier:	https://validator.w3.org/
Learn HTML:	https://www.w3schools.com/html/default.asp
Learn CSS:	https://www.w3schools.com/css/default.asp
Learn JavaScript:	https://www.w3schools.com/js/default.asp
Certification:	https://courses.w3schools.com/browse/certifications
Search Engine Optimization (SEO)	https://moz.com/learn/seo/what-is-seo https://www.spyfu.com/blog/meta-tags/

Appendix B

Columns, Classes, Internal CSS and Google Fonts

This is the code for two columns and a <nav> in the left column. Only the Homepage exists here, but the links for Pages 1-6 are included. CSS is internal in this webpage to show where to put internal CSS code. This webpage uses Google fonts that are linked to an external website in the <head> section. To view the fonts as intended, coders must be connected to the Internet when rendering in a browser. As mentioned in the Overview of this Guide, this section was created to show how classes in HTML and CSS are created and to show an alternative design. The rendered page for this website is on Page 7 (Appendix B). This code can be copied and pasted into Notepad, saved as “index.html” and opened in a browser. To edit the text in this webpage, right-click the “index.html” file, Open with, Notepad. Highlight the text between tags and edit. (Be careful not to delete an opening or closing tag.)

HTML5 and CSS Code:

```
<!DOCTYPE html>
 <html lang="en">

<head>
 <meta charset="UTF-8">
  <title>Title</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link href='https://fonts.googleapis.com/css?family=Julius Sans One' rel='stylesheet'>
   <link href='https://fonts.googleapis.com/css?family=Archivo Narrow' rel='stylesheet'>
   <link href='https://fonts.googleapis.com/css?family=Source Sans Pro' rel='stylesheet'>

<style>

article, aside, footer, header, main, nav, section {
  display: block;
}

html, body, ul, article, aside, footer, header, main, section {
  padding: 0px;
  margin: 0px;
}

header {
 margin-left: 220px;
}

.square {
  list-style-type: square;
}

.indent {
  margin-left: 30px;
  margin-right: 0px;
  float: left;
}
```

```
.column {
  float: left;
  padding: 10px;
}

.left {
  column-width: 200px;
  height: auto;
  font-family: 'Archivo Narrow';
  font-size: 25px;
}

.left:after {
  content: "";
  display: table;
  clear: both;
}

.right {
  width: 80%;
  height: auto;
}

h1 {
  font-family: 'Julius Sans One';
  font-variant-caps: titling-caps;
  text-align: left;
  margin-left: 0px;
  margin-bottom: 0px;
  font-size: 55px;
  color: #000000;
  padding: 0px;
}

h2 {
  font-family: 'Archivo Narrow';
  font-variant-caps: small-caps;
  font-size: 25px;
  color: #000000;
  padding-left: 5px;
  padding: 0px;

}

h3 {
  font-size: 18px;
  padding-left: 5px;
  padding: 0px;
}

p {
  padding-bottom: 0px;
  padding-top: 0px;
  padding-left: 0px;
}

body {
```

```
  font-family: 'Source Sans Pro';
  font-size: 20px;
  width: 100%;
  margin-left: 10px;
  margin-right: 10px;
  background-color: #ffffff;
  color: #000000;
}

nav {
  float: left;
  background-color: #ffffff;
  padding: 10x;
  margin-left: 10px;
  margin-right: auto;
  overflow: hidden;
  background-color: #ffffff;
  font-size: 22px;
}

ul {
  list-style-type: none;
}

li {
  float: left;
}

li a {
  display: block;
  color: #000066;
  text-align: center;
  padding: 15px 30px;
  text-decoration: none;
}

li a:hover {
  background-color: #f0f5f5;
}

a:link {
  color: #000066;
  margin-left: 0%;
}

a:visited {
  color: #3366cc;
}
a:hover {
  color: #8600b3;
}
a:active {
  color: #80b3ff;
}

* {
  box-sizing: border-box;
```

```
}

section {
  background-color: #ffffff;
  margin-top: 0px;
  padding-top: 0px;
  padding-bottom: 0px;
  padding-left: 0px;
  padding-right: 0px;
}

article {
  background-color: #ffffff;
  padding-top: 0px;
  padding-bottom: 0px;
  padding-left: 0px;
  padding-right: 0px;
  margin-right: 125px;
}

main {
  width: 100%;
  float: left;
  margin-bottom: 10px;
}

footer {
  clear: both;
  margin-left: 220px;
  background-color: #ffffff;
  color: #000000;
  font-family: 'Archivo Narrow';
  font-size: 22px;
  padding-top: 0px;
  padding-bottom: 50px;
  padding-left: 5px;
  padding-right: 5px;
}
 </style>

</head>

  <body>

   <header>
    <h1>Title</h1>
   </header>

<main>
 <section>
  <article>
  <div class="row">
  <div class="column left">
  <nav>
 <ul>
  <li><a href="index.html" target="_blank" rel="noopener noreferrer">Home</a></li>
  <li><a href="page2.html" target="_blank" rel="noopener noreferrer">Page 2</a></li>
```

```
 <li><a href="page3.html" target="_blank" rel="noopener noreferrer">Page 3</a></li>
 <li><a href="page4.html" target="_blank" rel="noopener noreferrer">Page 4</a></li>
 <li><a href="page5.html" target="_blank" rel="noopener noreferrer">Page 5</a></li>
 <li><a href="page6.html" target="_blank" rel="noopener noreferrer">Page 6</a></li>
</ul>
 </nav>
  </div>

<div class="column right">
 <article>
  <h2>Header 2</h2>
 <div>
  <ul>
   <li>List item.</li><br>
   <li>List item.</li><br>
   <li>List item.</li><br>
   <li>List item.</li><br>
  </ul>
 </div>
</article>

<article>
 <h2>Header 2</h2>
 <p>Body text.</p>
</article>
<article>
  <h2>Header 2</h2>
  <p>Body text.</p>
</article>
<article>
  <h2>Header 2</h2>
  <p>Body text.</p>
</article>
<div class="indent">
  <ul class="square">
   <li>List item with square bullets.</li><br>
   <li>List item with square bullets.</li><br>
   <li>List item with square bullets.</li><br>
  </ul>
 </div>
<br>
<br>
<br>
<article>
  <h2>Header 2</h2>
  <p>Body text.</p>
</article>
</div>
 </div>
  </article>
   </section>
    </main>
<footer>
<a href="page2.html">Next</a>
</footer>
</body>
</html>
```

Code Rendered in a Browser