1 INTRODUCTION TO WEB DESIGN

Module 1 Getting Started & HTML

Instructor Tonya D. Wright

File Management

VIEW PRESENTATION

At this point, we are ready to create the file structure that we will be using.

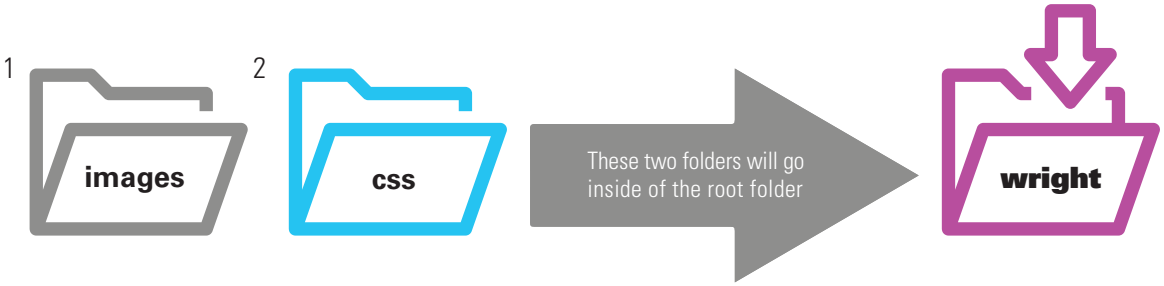
Let's start by creating the main folder for our site. This is usually referred to as the root directory because it is the first folder in the file tree of our Website. To create a new folder right-click on your mouse inside of your file browser or on your desktop and select the prompt you get for a new folder. Windows and Mac say different things.

Give your folder a name "the Website name or project name." for this course, we will be using your last name.

Create folder number 1, this will be our root folder

your last name (lowercase) Example: wright

Create two more folders



You will see why this basic order is very helpful to locate files for links as we move on. If you were using JavaScript a folder with js would work and so on. These are also common practices that would help another team member understand your order for development. This basic structure should be how you start development for all your Web projects, no matter what program you use for development.

An extra few tips:

Creating file folders can provide an added level of organization by grouping your files into categories. Give the folders clear and concise names. Names such as Miscellaneous and General can create ambiguity and make file searching arduous. On the other hand, specific names such as Images or Documents help you locate your files faster.

Nested folders

To further classify your files, and break them down to locate easily, you can create nested folders. Having multiple levels can help you track down the exact file you are looking for, and avoid the trouble of going through multiple files under one root folder. In the Images folder, you might have nested folders named Landing Pages, CTAs, Blogs, and Emails so that you can put pictures in the correct folder for which they were used.

While multi-level folders, or nested folders, you can add a level of sophistication to your organization; having too many levels can also become cumbersome. So, wherever possible, keep the nested folders to a minimum of two or three levels.

Put files where they belong

After you have created your folder structure, your next task is to put files in their respective folders. Be careful moving files later on as this will break the link that is made from the item to the Webpage.

"What's in a name?"

Every file you upload into your file manager has a name. But you can also add extra value assigning each file a unique title that will help make searching easier later. For example, if the image you uploaded is a picture of people strolling down the sidewalk, you might name it "walking."

Here are a few tips for naming files:

- Lowercase letters
- Use underscores between words (no spaces)
- Avoid special characters (!@#$%*)
- Shorter names are generally better, for longer names do not exceed 31 characters.
- Use a name that reflects the file/picture and what it is: img_parksunset.jpg logo_companyname.png

Create names that are logical, meaningful to all users, simple to read and relevant. These easy steps can help you stay organized and manage your files better, which in the end will save you time (and probably frustration).

PUBLISHING FILES TO A WEB SERVER

How would we put our files up on a Web server? I won't be covering this in class as there are so many different hosts and features.

Here is a link to help: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Upload_files_to_a_Web_server

WHAT'S A URL

With Hypertext and HTTP, URL is one of the key concepts of the Web. It is the mechanism used by Web browsers to retrieve any published resource on the Web. URL stands for Uniform Resource Locator. A URL is nothing more than the address of a given unique resource on the Web.

In theory, each valid URL points to a unique resource. Such resources can be an HTML page, a CSS document, an image, etc. In practice, there are some exceptions, the most common being a URL pointing to a resource that no longer exists or that has moved. As the resource represented by the URL and the URL itself are handled by the Web server, it is up to the owner of the Web server to carefully manage that resource and its associated URL.

```
1 https://developer.mozilla.org
2 https://developer.mozilla.org/en-US/docs/Learn/
3 https://developer.mozilla.org/en-US/search?q=URL
```

Any of those URLs can be typed into your browser's address bar to tell it to load the associated page (resource). A URL is composed of different parts, some mandatory and others optional. Let's see the most important parts using the following URL:

```
1 | http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#Sor
```

http:// is the protocol. It indicates which protocol the browser must use. Usually it is the HTTP protocol or its secured version, HTTPS.

The Web requires one of these two, but browsers also know how to handle other protocols such as mailto: (to open a mail client) or ftp: to handle file transfer, so don't be surprised if you see such protocols. (We use both of these in Web development)

www.example.com is the domain name. It indicates which Web server is being requested. Alternatively, it is possible to directly use an IP address, but because it is less convenient, it is not often used on the Web.

HOW TO USE URLS

Any URL can be typed right inside the browser's address bar to get to the resource behind it. But this is only the tip of the iceberg! The HTML language — which will be discussed later on — makes extensive use of URLs:

- to create links to other documents with the <a> element;
- to link a document with its related resources through various elements such as <link> or<script>;
- to display medias such as images (with the <img> element), videos (with the <video> element), sounds and music (with the <audio> element), etc.;
- to display other HTML documents with the <iframe> element.

Other technologies, such as CSS or JavaScript, use URLs extensively, and these are really the heart of the Web.

Absolute URLs vs relative URLs

The required parts of a URL depend to a great extent on the context in which the URL is used. In your browser's address bar, a URL doesn't have any context, so you must provide a full (or absolute) URL, like the ones we saw above. You don't need to include the protocol (the browser uses HTTP by default) or the port (which is only required when the targeted Web server is using some unusual port), but all the other parts of the URL are necessary.

When a URL is used within a document, such as in an HTML page, things are a bit different. Because the browser already has the document's own URL, it can use this information to fill in the missing parts of any URL available inside that document. We can differentiate between an absolute URL and a relative URL by looking only at the path part of the URL. If the path part of the URL starts with the "/" character, the browser will fetch that resource from the top root of the server, without reference to the context given by the current document.

Let's look at some examples to make this clearer.

Example of absolute URLs / Full URL's

```
https://developer.mozilla.org/en-US/docs/Learn
```

Example of relative URLs

Sub-resources

```
Skills/Infrastructure/Understanding_URLs
```

To better understand the example, let's assume that the URLs are called from within the document located at the following URL: https://developer.mozilla.org/en-US/docs/Learn

Because that URL does not start with /, the browser will attempt to find the document in a sub-directory of the one containing the current resource.

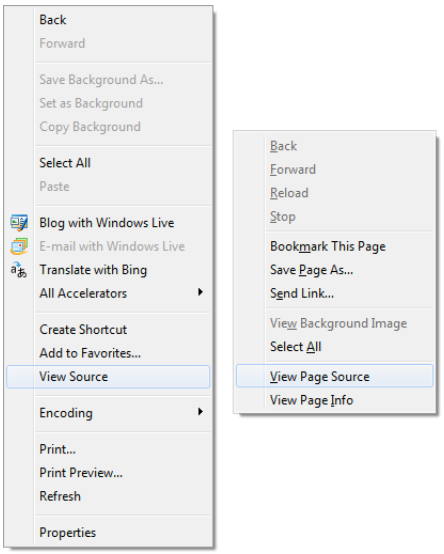
So in this example, we really want to reach this URL:
https://developer.mozilla.org/en-US/docs/Learn/Skills/Infrastructure/Understanding_URLs

Lost???

Hang in there with me, we will see this in action relative to what we do very soon. Just wanted to give you some ground rules on it.

Viewing Source Code

One of the great things about learning to build Web pages is that we all have the ability to view the source code of other people's Web pages. You can learn a lot by simply taking a peek at how another person's Web page was built … but how do you do that?



Although every browser uses slightly different terminology, the variations in how browsers let us view Web page code are so small that the process doesn't need to be spelled out for every browser. Here's the technique you'd use:

- Bring up a page in your browser, try this site https://www.w3.org/WAI/intro/accessibility.php

- Position your cursor on the page (not on an image but a blank spot), and right-click (or Ctrl-click if you're on a Mac). You should be presented with a context menu similar to what is shown here.

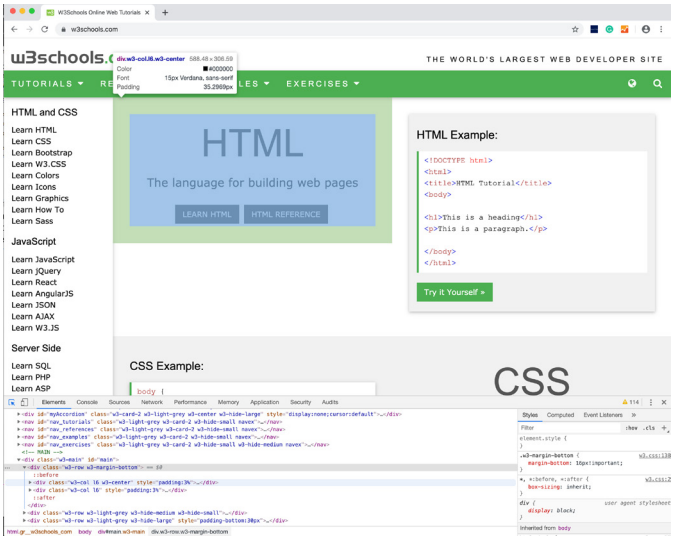
- Select View Source (View Page Source for Firefox), and a new window displays the page's underlying markup.

We won't analyze the markup, but this is one of those tricks that's really useful to know from the beginning.

Inspect Element

Another cool technique for working with pages is the inspect command. Let's use this link https://www.w3schools.com/ Right-click and select Inspect Element. We will use this in the course, each browser has this option layout and techniques may be a bit different. I have found the Inspect element to work best with Chrome.

Please note if you change options within inspect ,they are not real-time. Meaning, it will not automatically change/update your Website.



Here is a bit of help to walk you through the inspect element:

https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector/How_to/Open_the_Inspector

https://developer.mozilla.org/en-US/

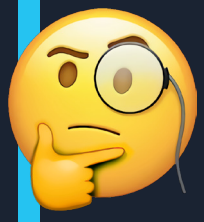
Introduction to HTML5

With HTML5, we can now embed audio and video natively within HTML documents. We can use inline SVG (Scalable Vector Graphics) markup. We can build more robust form experiences, complete with native error checking. We can create games, charts, and animations using the canvas element. Documents can communicate with each other using cross-document messaging.

In other words, HTML5 is much more of an application platform, not just a markup language.

Brackets will be our text editor of choice.

The formal definition: "A text editor is a type of program used for editing plain text files." Essentially, a text editor is a program on your computer that allows you to create and edit a range of programming language files. To the point - this is the place where you write your code! Text editors handle "hand coding" many different languages, i.e.: HTML, CSS, JavaScript, PHP, Ruby, Python, and so forth.



In Case You Were Wondering

Word Processors

Word Processors, including Microsoft Word, Pages, and even Google Docs, don't work as text editors because of their formatted text. We want plain text.

Important: Be careful copying and pasting from Word; the quotes are curly, not straight and will throw an error. I don't recommend copying and pasting from Word.

Below is a link to a list of text editors, just in case. Don't forget we already have a text editor with Dreamweaver. One free and robust text editor I also use is Brackets - http://brackets.io/

Review best text editors
http://www.creativebloq.com/netmag/10-great-text-editors-Web-designers-71412411

What is HTML?

HTML is the standard markup language for creating Web pages.

HTML stands for HyperText Markup Language
HTML describes the **structure** of a Web page
HTML consists of a series of elements
HTML elements tell the browser how to display the content
HTML elements are represented by tags
HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
Browsers do not display the HTML tags but use them to render the content of the page

```
<!DOCTYPE html>
<html lang="en">
<html>
<head>
<meta charset="UTF-8">
<title>Untitled Document</title>
</head>

<body>

    <!--Web Content Goes Here-->


</body>
</html>
```

CODE EXPLAINED

The **<!DOCTYPE html>** declaration defines this document to be HTML5
The **<html lang>** declares the language of a Web page or portion
The **<html>** element is the root element of an HTML page
The **<head>** element contains meta information about the document
The **<title>** element specifies a title for the document
The **<body>** element contains the visible page content
The **<h1>** element defines a large heading
The **<p>** element defines a paragraph

Note about the doctype: The <!DOCTYPE> declaration represents the document type and helps browsers to display Web pages correctly. It must only appear once at the top of the page (before any HTML tags). The <!DOCTYPE> declaration is not case sensitive. The <!DOCTYPE> declaration for HTML5 is: <!DOCTYPE html>

Note about HTML lang: The HTML tag tells the browser that this is an HTML document, with an attribute of the language being in English. Despite the attribute, this is an opening tag for the HTML element.

Note about meta: You may have spotted the <meta> tag in the markup above. We have not covered this yet, although the <meta> tag is not part of the skeletal requirements of a Web page, it has a useful purpose, especially supporting information about the Web page.

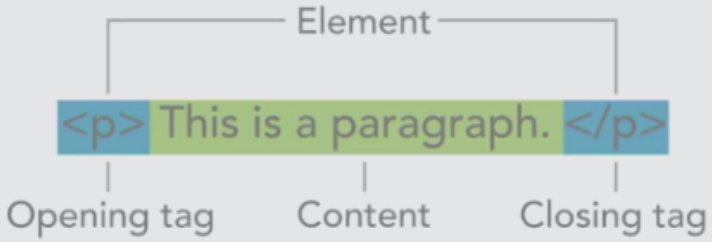
Semantic Web Design

HTML was initially designed as a presentation language. In other words, it was intended to display technical documents in a browser in a readable and predictable manner. The element list in the first version of HTML explained how the content would be displayed. You would have probably seen a whole page full of div's, h tags, and p tags. HTML5 has added a significant number of new tags to help add semantic meaning to our code. Tags such as <header>, <footer>, <article> and <section>. To me, I see this as an efficiency boost. As I am reviewing code, I can quickly pick up from these tags, where they are in the visual design. Semantic Web design is a great move for clean and organized coding.

Let's talk about tags

An HTML document is built using elements. Remember, elements are the bricks that create the structures that hold a Web page together. But what exactly is an element?

So it looks something like this: This Is My First Paragraph (by the way **p** stands for paragraph)



- HTML tags *normally* come in pairs like and
- The first tag in a pair is the opening tag,
- The second tag is the closing tag,
- The closing tag is written like the opening tag, but with a slash "/" inserted before the tag name.

TAGS, ELEMENTS, AND ATTRIBUTES

A lot of people tend to confuse the terms tag and element.

tag refer to the actual tags themselves, example:

element is the tag plus its contents, example: This is a paragraph

attribute

HTML elements can have a range of attributes, depending on which element you're dealing with. Each attribute is made up of a name and a value, and these are always written as name="value".

Some attributes are optional, but together they give the browser important information that the element wouldn't otherwise offer. For example, the image element (which we will learn about soon) has a compulsory "image source" attribute, the value of which gives the filename of the image. Attributes appear only in the opening tag of any given element.

You can also pass on more information about elements by using attributes, just like in the opening HTML tag, which has a language attribute. Some attributes are used to give the necessary information:

Such as where to find an image	<img src="images/poppy.jpg">
Where a link should point to	This is a link.
Help give the element meaning; such as the class attribute which further identifies the paragraph content.	Author: James Williamson

Frequently used HTML tags

Below are some of the most commonly used HTML tags. To get the most out of coding, it helps to understand the nature of these elements and how they are used. Remember that some tags can serve multiple purposes.

TAG	DESCRIPTION
<!--...-->	Comment. Designates an HTML comment. Allows you to add notes within the HTML code (represented by ... in the tag) that are not displayed within the browser.
<a>	Anchor. The basic building block for a hyperlink.
<blockquote>	Quotation. Creates a standalone, indented paragraph identifying content quoted from another source.
<body>	Body. Designates the document body. Contains the visible portions of the Webpage content.
 	Break. Inserts a visual line break without creating a new paragraph.
	Division. Used to divide Webpage content into discernible sections.
	Emphasis. Adds semantic emphasis. Displays as italics by default in most browsers and readers.
<form>	Form. Designates an HTML form. Used for collecting data from visitors.
<h1> to <h6>	Headings. Creates headings. Default formatting is bold.
<head>	Head. Designates the document head. Contains code that performs background functions, such as meta tags, scripts, styling, links, and other information not overtly visible to site visitors that may provide instructions on how to display the page or its contents.
<hr>	Horizontal rule. Empty element that generates a horizontal line.
<html>	Root element of most Webpages. Contains the entire Webpage, except in certain instances where server-based code must load before the opening <html> tag.
<iframe>	Inline frame. A structural element that can contain another document or load content from another Website.
<img>	Image. Provides the source reference to display an image.
<input>	Input. An input element for a form such as a text field.
	List item. An element within an HTML list.
<link>	Link. Designates the relationship between a document and an external resource.
<meta>	Metadata. Additional information provided for search engines or other applications.
	Ordered list. Defines a numbered list. List items display in an alpha, numeric, or roman numeral sequence.
<p>	Paragraph. Designates a standalone paragraph.
<script>	Script. Contains scripting elements or points to an internal or external script.
	Span. Designates a section within an element. Provides a means to apply special formatting or emphasis to a portion of an element.
	Strong. Adds semantic emphasis. Displays as bold by default in most browsers and readers.
<style>	Style. Embedded or inline element or attribute containing CSS styling.
<table>	Table. Designates an HTML table.
<td>	Table data. Designates a table cell.
<textarea>	Text area. Designates a multi-line text input element for a form.
<th>	Table header. Identifies a table cell containing a header.
<title>	Title. Contains the metadata title reference for the current page.
<tr>	Table row. Structural element that delineates one row of a table from another.
	Unordered list. Defines a bulleted list. List items display with bullets by default.

Frequently used HTML character entities

Entities are frequently used to display reserved characters (which would otherwise be interpreted as HTML code), and invisible characters (like non-breaking spaces). You can also use them in place of other characters that are difficult to type with a standard keyboard.

CHARACTER	DESCRIPTION	NAME
©	Copyright	©
®	Registered trademark	®
™	Trademark	
•	Bullet	
–	En dash	
—	Em dash	
	Nonbreaking space	
&	Ampersand	&

Go to www.w3schools.com/html/html_entities.asp to see a complete list and description of entities.

New tags for HTML5

Below are some of the important new tags in HTML5. The specification features nearly 50 new tags in total, while at least 30 old tags were deprecated. Take a few moments to familiarize yourself with these tags and their descriptions.

Go to www.w3schools.com/tags/default.asp to see the complete list of HTML5 elements.

TAG	DESCRIPTION
<article>	Article. Designates independent, self-contained content that can be distributed independently from the rest of the page or site.
<aside>	Aside. Designates sidebar content that is related to the surrounding content.
<audio>	Audio. Designates multimedia content, sounds, music, or other audio streams.
<canvas>	Canvas. Designates graphics content created using a script.
<figure>	Figure. Designates a section of standalone content containing an illustration, image, or video.
<figcaption>	Figure caption. Designates a caption for a <figure> element.
<footer>	Footer. Designates a footer of a document or section.
<header>	Header. Designates a section of the content that introduces a document or specific topic area.
<hgroup>	Heading group. Designates a set of <h1> to <h6> elements when a heading has multiple levels.
<main>	Main. Designates the unique content of a page. A page may have only one main element.
<nav>	Navigation. Designates a section containing a navigation menu or hyperlink group.
<picture>	Picture. Designates one or more resources for a Webpage image to support the various resolutions available on smartphones and other mobile devices. This is a new tag that may not be supported in older browsers or devices.
<section>	Section. Designates a section within the content of a document.
<source>	Source. Designates media resources for video or audio elements. Multiple sources can be defined for browsers that do not support the default file type.
<video>	Video. Designates video content, such as a movie clip or other video streams.

HTML DEMO: <h1> – <h6>

On the list for frequently used tags we saw <h1> – <h6>, these elements represent six levels of section headings. <h1> is the highest section level and <h6> is the lowest. Below is a breakdown to show you how they look. As the chart stated, the bold attribute is by default. So you will never need to add strong tags to any of these elements.

CODE

```
<h1>The Greatest Title of All Time</h1>
    <h2>What do you think can go next?</h2>
        <h3>Pretty handy Title</h3>
            <h4>Golf may actually be fun</h4>
                <h5>Would you ride a bike or drive a car</h5>
                    <h6>Wee little last title here</h6>
```

OUTPUT

The Greatest Title of All Time

What do you think can go next?

Pretty handy Title

Golf may actually be fun

Would you ride a bike or drive a car

Wee little last title here

Search engines use the headings to index the structure and content of your Web pages. Users often skim a page by its headings. It is important to use headings to show the document structure.

<h1> headings should be used for main headings, followed by <h2> headings, then the less important <h3>, and so on.

Each HTML heading has a default size. However, you can specify the size for any heading with the style attribute, using the CSS font-size property.

What you see

The purpose of a Web browser (Chrome, Edge, Firefox, Safari) is to read HTML documents and display them. The browser does not display the HTML tags, but uses them to determine how to display the document:

As shown in the example, the tags and comments don't show up. Yep! All that code to show one line.

MODULE 1 RESOURCES

HOW TO PUBLISH YOUR FILES TO A WEB SERVER:
https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Upload_files_to_a_Web_server

LEADING THE WEB TO ITS FULL POTENTIAL
https://www.w3schools.com/

INSPECTOR
https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector/How_to/Open_the_Inspector

MDN WEB DOCS
https://developer.mozilla.org/en-US/

https://developer.mozilla.org/en-US/docs/Web/HTML

https://developer.mozilla.org/en-US/docs/Web/HTML/Element

https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes

DEVELOPMENT HELP
www.w3schools.com/tags/default.asp

https://html.spec.whatwg.org/multipage/

https://webplatform.github.io/

DESIGN TOOLS
https://mybrandnewlogo.com/color-gradient-generator

https://calcolor.co/

https://thedesignteam.io/about-being-colorblind-bc0a58bcdfc2