# Chapter 22: Web Writing

## Objectives

Upon completion of this chapter, readers will be able to

1. Apply the fundamentals of HTML, including its purpose, structure, and basic rules for using tags and attributes.
2. Identify and apply common HTML tags for structuring web page content, such as paragraphs, headings, lists, and links.
3. Embed external content, such as YouTube videos, into a web page and modify its attributes.

## Introduction to HTML

HTML (Hypertext Markup Language) is a markup language that makes up web pages. Did you know that HTML is a language with rules, just as English is?  HTML is a coded computer language that, when read by a web browser, displays the web pages you see every day when exploring the Internet.

See for yourself: In Mozilla Firefox, right-click on a web page and click "Inspect Element." The code of the page you are looking at should appear at the bottom of the page. This code is what Firefox reads to show you the page you are seeing in your browser.

Right-Click on a PC

CTRL+Click on a Mac

HTML is the baseline coding of every web page you see out there, and when combined with CSS, JavaScript, and other languages, it can create cleanly-constructed and functional websites.

**CSS** - Cascading Style Sheet. This language is an advanced styling language that can be written into the HTML document or linked to from a separate CSS document.

**JavaScript** - High complexity programming language that makes the more complicated things you see on some websites, like games.

### Tag Rules

HTML is a language, and it, as all languages do, has rules that make it make sense to the web browser. Just as the English language doesn't make sense when you break grammar rules, neither does HTML when you break tag rules. The difference is that most English speakers will still understand what you are trying to say when you break a grammar rule; the web browser will not understand when you break a tag rule. So, let's look at some of these rules.

When you look at a line of HTML, you'll see information enclosed in brackets **< >** throughout the code. Brackets denote a tag. A tag tells web browsers what each item on the page is; it tells the web browser how to classify that information. For example, a paragraph would be classified by a **<p>** tag.

#### Rule #1

*Most* tags have an opening tag and a closing tag that surround the information it is classifying or effecting. An opening tag is just the brackets with the tag enclosed, like so: **<p>**. A closing tag is the same thing, except there is a forward slash at the front of the tag, like so: **</p>**. When tagging a paragraph with the **<p>** tag, it would look like the example.

**<p>**This is a paragraph about how to properly tag a paragraph. To tag a paragraph, you have to start with an opening paragraph tag, type the paragraph, and then end with a closing paragraph tag. This is how most sections of simple paragraph text are tagged.**</p>**

#### Rule #2

Rule #1 does not *always* apply. Not all tags have to have a closing tag, and you just have to memorize those. We will learn some basic tags in the next section, where we talk about a few that do not require a closing tag.

#### Rule #3

Opening and closing tags must be *nested* to work. That is, the last tag you open should be the first tag you close, as shown in the example.

**<body>**

**<p>**This is a paragraph in the body of the webpage. As you see, I opened the body tag first, and then the paragraph tag; if I am nesting the tags, then I have to close the paragraph tag first, and then close the body tag. **</p>**

**</body>**

### Basic Tags

There are numerous HTML tags, making it impractical to memorize them all. Fortunately, the Internet provides easy reference, so memorization is unnecessary. This section introduces the essential tags commonly used in nearly every web page.

#### Beginning Tags and the <head>

Every HTML web page starts with a **<!doctype html>** tag, with no closing tag. This is what tells the web browser what version of HTML the page is written in. The **<!doctype>** tag used to have several options in the last version of HTML, but in HTML5, there is only one: **<!doctype html>**.

After the **<!doctype html>** tag, there is an **<html>** tag, which is also required for every HTML document. Unlike the **<!doctype html>** tag, the **<html>** tag does require a closing tag of **</html>**. For nesting purposes, the closing **</html>** tag will always be the very last thing in your document.

Once the **<html>** tag has been opened, the page then has its head information in the **<head>** tag, which also has a closing tag of **</head>**. The **<head>** tag includes information that does not show up on the page but is vital to the look and feel of the page. There is always a **<title>** tag with a closing **</title>** tag that encloses the information on the tab of the web browser. Users who are including CSS information in their document might also include CSS coding in the **<head>** tag.

Figure 1: A browser tab with information

*Figure 1: A browser tab with title information*

#### The <body> Tag and Other Common Tags

After the **<head>** tag has been closed, the next tag to open is the **<body>** tag. This denotes the information that you see on the page when looking at a web browser. Everything you see within the browser window appears after the **<body>** tag and before the closing **</body>** tag.

For common tags used within the **<body>** tag and their purpose, see Table 1. Unless otherwise noted, all tags in the table below require a closing tag.

*Table 1: Common HTML Tags*

| **Tag** | **Function** |
| --- | --- |
| <em> | Defines emphasized text. Text enclosed in an **<em>** tag will appear italicized. |
| <strong> | Defines important text. Text enclosed in a **<strong>** tag will appear boldface. |
| <img> | Defines an image. This tag requires an attribute that defines what image it shows. This attribute is called **src**. The **<img>** tag does not require a closing tag; however, it does end differently. Instead of the usual closing tag, an **<img>** tag is closed by ending the opening tag with a space and then **/>**. For example:  **<img src="logo.jpg" />** |
| <a> | Defines a link. This tag requires an attribute that defines where the link goes to. This attribute is called **href**. |
| <ul> | Defines an unordered list. This tag will start a bulleted list. |
| <ol> | Defines an ordered list. This tag will start a numbered list. |
| <li> | Defines a line in a list. This tag applies to both unordered and ordered lists. When you start a list, you must use **<li>** tags to open and close each line separately. See Figure 4 for an example of how a list looks in HTML. |
| <table> | Defines a table. This tag will start a table. |
| <tr> | Defines a table row. This tag will start a row in a table. You must have a **<tr>** tag for each row of a table. |
| <td> | Defines a table column. This tag will start a column in a row of a table. You must have a new **<td>** tag for each column of each row (each square) of the table. See Figure 5 for an example of how a table looks in HTML. |
| <br> | Defines a line break. This tag does not require a closing tag. |
| <h1> - <h6> | Defines headings. There are six HTML heading levels: **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>**, and **<h6>**. They each define headings of their specified order (**<h1>** being the first order heading and **<h6>** the sixth order). |
| <p> | Defines a paragraph. This is how most sections of text are tagged. |

*Table 2: Ordered and Unordered Lists Preview*

| **HTML** | **Preview** |
| --- | --- |
| <ul>  <li>Item 1</li>  <li>Item 2</li>  <li>Item 3</li>  </ul> | Item 1  Item 2  Item 3 |
| <ol>  <li>Item 1</li>  <li>Item 2</li>  <li>Item 3</li>  </ol> | Item 1  Item 2  Item 3 |

*Table 3: Table Preview*

| **HTML** | **Preview** |
| --- | --- |
| <table>  <tr>  <td>Box 1</td>  <td>Box 2</td>  <td>Box 3</td>  </tr>  <tr>  <td>Box 4</td>  <td>Box 5</td>  <td>Box 6</td>  </tr>  </table> | |  |  |  | | --- | --- | --- | | Box 1 | Box 2 | Box 3 | | Box 4 | Box 5 | Box 6 | |

### Basic Attributes

HTML attributes provide more information for tags or apply stylistic features to them that are not the default. For example, an **<a>** tag requires more information to know where it should link to, and a **<p>** tag can have a different font color than the default if you want it to. Let's look at some common attributes and what they do in Table 2 below.

*Table 4: Common HTML Attributes*

| **Attribute** | **What It Does** | **What It Looks Like in HTML** |
| --- | --- | --- |
| href | The **href** attribute belongs to the **<a>** tag and defines where the link will go to. | <a href=""http://distanceed.hss.kennesaw.edu/technicalcommunication">This is a link to the textbook website.</a> |
| src | The **src** attribute belongs to the **<img>** tag and defines where the tag will pull the image from. This can be an image site or a file path from within the server. | <img src="logo.jpg" /> |
| width | The **width** attribute can be applied to images, iframe elements, tables, or other elements with a numerical width. It defines how wide the item will be. | <img src="logo.jpg" width="300px" /> |
| height | The **height** attribute can be applied to images, iframe elements, tables, or other elements with a numerical height. It defines how tall the item will be. | <img src="logo.jpg" width="300px" height="300px" /> |
| style | The **style** attribute allows you to add inline CSS elements to your tag. A few common ones are **color**, **font-size**, **font-family,** and **text-align**. You define a style element the same way as other attributes, however you then must define the value of the style element. View the examples to the right to see how these style elements are defined. | <p style="color:blue;"> This is blue text.</p>  <h1 style="font-size:24px;">This is a 24pt font heading.</h1>  <a href="http://distanceed.hss.kennesaw.edu/technicalcommunication" style="font-family:Arial; font-size:12px;">This link to the textbook website is Arial 12 pt font.</a>  <h2 style="text-align:center;">This heading is centered on the page.</h2> |

### Embeddable YouTube Videos

YouTube remains the most widely used video-sharing platform in the world, and it’s easy to add its videos to your own website or online portfolio using **embed codes**. Embedding allows you to display content—like a video—directly on your web page without uploading the file yourself. Instead, the video plays from YouTube inside a special HTML element called an <iframe>, which acts like a window into the original site. You can usually find the embed code by clicking the **Share** button under the video and selecting **Embed**. The best part? You don’t need to modify the code—just copy and paste it into your HTML. YouTube even offers a **privacy-enhanced mode** that reduces tracking.

To make the video responsive (so it scales properly on phones and tablets), use this HTML and CSS together:

| **HTML** | **CSS** |
| --- | --- |
| <div class="video-container">  <iframe src=https://www.youtube-nocookie.com/embed/VIDEO\_ID title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>  </div>  *Just replace VIDEO\_ID with the actual ID of the YouTube video you want to show (the part after watch?v= in a YouTube URL).* | .video-container {  position: relative;  padding-bottom: 56.25%; /\* 16:9 aspect ratio \*/  height: 0;  overflow: hidden;  max-width: 100%;  }  .video-container iframe {  position: absolute;  top: 0;  left: 0;  width: 100%;  height: 100%;  } |

This approach ensures your video looks good on desktops, tablets, and phones alike.

Also, you can edit code. Let's analyze a YouTube embed code.

<iframe width="420" height="315" src="https://www.youtube.com/embed/FqzDgjlN4GY?si=pN1NYJSfE-RDh4Zb" frameborder="0" allowfullscreen></iframe>

**iframe** - Most embed codes use the iframe tag now, but not all of them. It depends on the website and what kind of code was used to create the item.

**width=”420” height=”315”** – You can edit the width and height of embed codes or add them in if it is not already in the code. This is good if the default size is not quite big or small enough for what you want to use it for. Just change the number in the quotations to change it. The default measurement here is pixels, but you can also change it by adding a different measurement. For example, setting width=”80%” would tell the browser to display the video on 80% of the width of the window.

**src="https://www.youtube.com/embed/FqzDgjlN4GY?si=pN1NYJSfE-RDh4Zb"** - The source of embed codes are usually some variation of the link to the item. If you put this link in your web address, it might not work (it also might just give you a full-screen version of the video). But if you take out the "embed/" part of it and replace it with "watch?v=", it will take you to a video on YouTube of one of my old projects from grad school. Go ahead, try it.

Watch the video below to see tags, attributes, and embed codes in action!

[HTML in Action](https://youtu.be/ftop-1fVSws)

### Organizing Your Web Page

Web pages are a series of pages connected by hyperlinks, and they have unique needs. Let's think about it this way: you have a small child whom you keep in a playpen. In the playpen is a tiny toy box with a ball, a block, and a fire truck. You play a game where you tell her to show you the ball, and she points to the ball. You tell her to show you the block, and she points to the block. You tell her to point to the fire truck, and she points to the fire truck. This game works very well.

But then you take her into another room and put her in another playpen, and there's a pillow, a rattle, and a dolly. You tell her to point to the fire truck, and she looks around and shrugs her shoulders. Her mind is not advanced enough to understand that you wish her to point through the wall into the other playpen. She does not have the physical ability to get out of the playpen, walk to the other room, and retrieve your fire truck for you. This game will not work.

You then try again. You put her back in the original playpen, but you take the small box of toys and scatter them around the house. You put the ball under the couch. You put the block in the bathtub. And you put the fire truck in the trunk of your car. You return to the playpen and ask the child to show you the fire truck. Again, she shrugs her shoulders. This game will not work.

Why does the first game work, but the other two do not?

What if you change the game by taking the child, playpen, toy box, and all into the next room and again ask the child for the block? Unless something distracts the child, the game will work! The child knows the block is in the box, and all functions as normal.

Your web page is the child, and its name is index. Your main folder is the playpen, and your subfolders are the toy boxes.

When you make the homepage of a website, you'll want to name it "index.html" and put it in a folder. In that folder, you can then put any pictures, documents, videos, or other HTML files that you might link to. You can also organize further by putting those things into subfolders. But no matter what, everything that is not already online must be put inside that main folder to be then posted online.

### Where to Practice and Learn More HTML Skills

[Codecademy](http://codecademy.com/)

[W3](http://w3schools.com/)[S](http://w3schools.com/)[chools](http://w3schools.com/)

[Khan Academy](https://www.khanacademy.org/computing/computer-programming/html-css)

## General Web Design Principles

Websites' layout, design, purpose, and function vary tremendously, but most websites follow a set of basic concepts that you can use to evaluate and edit them. Consider these ideas when you edit websites:

* Are words or phrases used consistently across the whole website? Are these words meaningful to the users? If not, what words would the readers prefer or understand?
* Are related activities close to each other onscreen and in a logical location? If not, where would the user like to see those activities?
* Do the pages have plenty of headings? If not, where would headings be useful and appropriate?
* Are the sentences and paragraphs short? If not, how and where could you break up long blocks of text into bite-sized chunks?
* Are there any paragraphs that can become lists?
* Are the typefaces and font sizes appropriate for use onscreen? If they're hard to read, how could they be improved?
* Are the hyperlinks easily visible?
* Do the hyperlinks' visible text and mouse-over text describe the items that will open? (In particular, rename links that say "click here.")
* Do all hyperlinks work?
* Do the navigation menus appear in the same location and look identical on every page?
* Do the navigation menus work?
* Does the website include a sitemap?
* Does the website have an internal search engine? Does it work?
* Is the website usable on a mobile device?
* Does the onscreen layout follow an F-pattern? (Most people read websites in this pattern: they look across the top, scan down the left side a few inches, read across, and scan down again.)
* How long would it take for a user to realize the site is ineffective in some way?
* Can the user start over if a problem occurs?

Also, just final a note, websites must now include privacy information, cookie consent, PII info and data storage…basically a policy page. [Elementor blog](https://elementor.com/blog/website-legal-requirements/) has information about website legal requirements.

## Attribution

This chapter is revised from the first edition of *Open Technical Communication*, Chapter 9: “[HTML Basics](https://alg.manifoldapp.org/read/open-technical-communication/section/48d87993-3fe9-452f-9ca4-f03c9713afde)” by Tiffani Tijerina and Chapter 8: “[Technical Editing](https://alg.manifoldapp.org/read/open-technical-communication/section/19abea6b-932a-4efd-a70d-9f1123dd4b08)” by Jonathan Arnett, which are both openly available under a Creative Commons Attribution license.

## AI Assistance Notice

Some parts of this chapter were brainstormed, drafted, and/or revised in conversation with ChatGPT 4o and Google Gemini 2.5 Flash. All AI-generated content was reviewed and revised as needed by a human author.