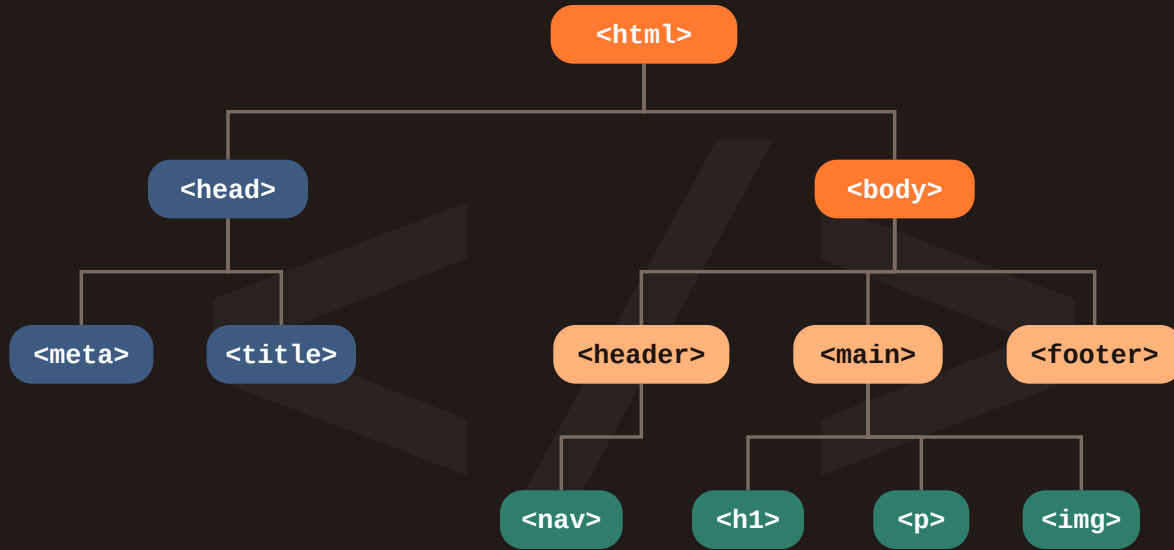




BioGlytch Studios LLC

Learn | Build | Grow



Every web page is a tree of elements

WEB DESIGN WITH HTML5

BUILD YOUR OWN PORTFOLIO

WRITTEN BY

Clarence Scott & BioGlytch Studios

BioGlytch Studios LLC | Est. 2026

In Partnership with Gloria Coles Flint Public Library

bioglytchstudios.tech | 2026 Edition

Web Design with HTML5

HTML Fundamentals

BioGlytch Studios LLC

COPYRIGHT 2026 © BioGlytch Studios LLC © Genesee County, MI EST 2026

</HELLO CODER>

Welcome to the engineering world of BioGlytch Studios LLC! We are thrilled to present this book to you. Published by BioGlytch Studios LLC and offered as a front-facing resource of Gloria Coles Flint Public Library, it is dedicated to making computer science and technology education more accessible to our community. Designed for aspiring technology enthusiasts and their supportive parents, this book offers an opportunity to explore the exciting world of computing together. If you are a parent who has chosen this book to introduce your child to technology, we thank you for your trust and support.

Through this book, and our partnership with Gloria Coles Flint Public Library, BioGlytch Studios LLC aims to ignite curiosity and inspire a passion for coding, programming, and computing technologies in individuals of all ages. We recognize that access to quality educational resources can be limited by financial barriers, time constraints, and the rising costs of traditional university courses. Our goal goes beyond teaching the fundamentals—we want to empower you to think critically, solve problems creatively, and begin your journey toward becoming a skilled and innovative engineer.

Throughout this book, you will develop essential skills such as critical thinking, logical outlooks, problem solving, discipline and career-ready structure. Together, we are committed to building a stronger, more technologically empowered community, one coder at a time.

Each successful assessment will grant a Certificate from BioGlytch Studios LLC, along with a PDF of your course completion to show your next employer.

Thank you for joining us on this journey. We look forward to supporting your growth and celebrating your progress as you put your best code forward!

Sincerely,

Clarence Scott

CEO & Founder, BioGlytch Studios LLC

In Partnership with Gloria Coles Flint Public Library

Publishing Information

Book Title

Web Design with HTML5

Author

Clarence Scott

Publisher

BioGlytch Studios LLC

2026 Edition

Copyright Notice

© 2026 BioGlytch Studios LLC. All rights reserved.

BioGlytch Studios LLC is the sole owner of all content in this book. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the prior written permission of the publisher, except in the case of brief quotations embodied in critical articles or reviews.

Library Partnership

Gloria Coles Flint Public Library offers this book as a front-facing resource for its organization. The partnership does not transfer ownership of any content, which remains solely with BioGlytch Studios LLC.

Edition

First Edition

Disclaimer

The information contained in this book is provided for educational purposes only. While every effort has been made to ensure accuracy, the author and publisher are not responsible for errors, omissions, or damages resulting from the use of this material.

Printing Information

Printed in Genesee County, Michigan

Printed by BioGlytch Studios LLC

Contact Information

For inquiries, feedback, or bulk orders, please contact:

Email: software@bioglytchstudios.tech

Website: <https://bioglytchstudios.tech>

Phone: (989) 900-6747

Table of Contents

Web Design with HTML5: Beginner's Guide

- Introduction

Chapter 1: Getting Started with HTML

- Creating Our Docs “Hello World”

Chapter 2: Headings and Comments

- Headings
- Comments

Chapter 3: Elements, Paragraphs and Attributes

- Elements
- Paragraphs
- Attributes

Chapter 4: Images, Lists and Tables

- Images
- Lists
- Tables

Chapter 5: Links, Block and Inline Elements

- Links
- Block Elements and Div Containers
- Inline Elements

Chapter 6: Forms

- GET vs. POST
- Input Types, Placeholder and Required
- Form-Building Tools

Chapter 7: Colors, Audio and Video

- HTML Colors
- Audio and Video Elements

2026 Best Practices for HTML

Conclusion

- Recap of Key Concepts
- Encouragement to Explore and Experiment

Appendix

- Comprehensive Glossary - Section A (Attributes, Elements & Page Structure)

- Comprehensive Glossary - Section B (Forms, Publishing & Web Basics)

Useful Websites and Tools

- Tools for Writing and Testing Code
- Design Resources
- Learning and Reference
- Technical Support and Communities

Web Design with HTML5: Beginner's Guide

Introduction

Our Web Design Goals

- **Plan and Structure:** Learn how to plan your web design and structure your flow chart. Understand your goal and purpose.
- **Write HTML Markup:** Learn our text editor, create files, and implement code (known as markup in HTML) to build our webpage.
- **Apply Styling with CSS:** Using inline cascading styling, learn about external files and connecting them to HTML files.
- **Test and Debug:** Preview the web page to ensure it displays as intended. Check for any errors or inconsistencies in layout and functionality; these are known as “bugs in code.” We’ll fix these.
- **Publishing and Maintenance:** Upload your HTML file, and any files in the web page’s folder, to a web server on the internet or an intranet. For this book you will be creating a portfolio to upload.

Certificates

Each successful assessment will grant a Certificate from BioGlytch Studios LLC, along with a PDF of your course completion to show your next employer.

Overview

HTML5 is a markup language used for structuring and presenting content on the World Wide Web (WWW). This is the fifth version of the HyperText Markup Language. In other words, HTML is what you see today in your browser, or “internet,” when you are using your computing devices. HTML is the center of every web page. Behind everything you search on the web is mainly HTML or some form of HTML code. It is used so your browser can display text, images and tables.

In this book you’ll work with HTML code to control the structure and elements of your web page. Browsers look for HTML documents to display the webpage, and a webpage can include many types of elements. Now that you have a little more information on what we will be doing, let’s begin building your very own web page.

Pre-Requisites

In order to begin coding with us you only need 3 things: a computer (with monitor), a text editor, and the drive to want to code. If you do not have a text editor, we will walk through downloading one. For this book we will use Visual Studio Code, developed by Microsoft.

1 Download Visual Studio Code. Head over to Microsoft's website and download Visual Studio Code. In the Extensions tab, install the extension **Live Server** by Ritwick Dey and you will be ready to continue. If you need help, refer to our YouTube video "Setting Up Visual Studio for Development."

2 Find a quiet setting. Make sure you can focus on coding and studying the information in this book. Focus is key, determination is power. Code is the goal.

3 Grab a snack and something to drink (always keep water near you) so you can stay mentally strong while studying and coding. Your brain will become tired and mentally drained if you do not.

Chapter 1: Getting Started with HTML

Creating Our Docs “Hello World”

Now that you are ready to code, let’s take a look at our setup in Visual Studio Code. If everything was done correctly, your screen should show the Visual Studio Code welcome window. If so, you have successfully downloaded Visual Studio Code.

Hover over the “Open Editors” section and add a new file. Do that. In that file we are going to add the text “Hello World” and save the file. When the popup shows to name your file, create a folder where you are and name the folder “Portfolio,” then name your text file “index.html” and save it.

Reopen Visual Studio Code with this new folder, either by closing the app and opening it from the folder path (right-click) or by clicking the “File” tab in the Visual Studio Code window and proceeding from there.

Assuming you have installed the Live Server extension, you should see “**Go Live**” in the bottom right-hand corner. Click it and watch what happens next. You should have your webpage showing! It will say “Hello World,” just as you put in the code.

Now that we have our HTML doc created, let’s dig deeper. Replace “Hello World” with the text **html:5** . When the drop-down shows, press Enter. Your page should now contain the HTML5 boilerplate:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

</body>
</html>
```

Practical Exercises: Chapter 1

- Exercise 1.1 - Install and Launch:** Install Visual Studio Code and the Live Server extension, then confirm the “Go Live” button appears.
- Exercise 1.2 - Hello World:** Create a folder named “Portfolio” with a file named index.html containing “Hello World,” and view it with Live Server.
- Exercise 1.3 - Boilerplate:** Replace “Hello World” with html:5, press Enter, and review the generated boilerplate.

Chapter 2: Headings and Comments

Headings

HTML includes six levels of headings, ranked from 1 to 6: <h1> through <h6> . To use these tags properly we need to make sure we close out the element's tag. If you use the H1 tag, the proper way to use it is:

```
<h1>Hello World</h1>
```

Inside the body tag, place an H1 and an H3 element. In the first, put your name. In the second, put your new career title, "Web Designer."

```
<h1>BioGlytch Studios LLC</h1>  
<h3>Web Designer</h3>
```

Comments

Once you see your headings reflected in the browser, let's add a comment. The cool thing about comments is that they do not affect your code when done properly, so you can add as many as you want in any coding language. Comments are powerful because they help you and other developers understand your code and where your thinking was at with execution.

A comment is opened with an angle bracket, an exclamation point and two hyphens, and closed with two hyphens and an angle bracket. Similar to elements, above your H1 element, add a comment to let someone know this is the header for your webpage:

```
<!-- This is the header for my webpage -->  
<h1>BioGlytch Studios LLC</h1>  
<h3>Web Designer</h3>
```

After you add your code and refresh your browser, you should notice that your comment is not showing on the page. Congrats! You have successfully added a comment for the next developer to know where your heading section is.

Practical Exercises: Chapter 2

1 Exercise 2.1 - Heading Levels: Add an H1 with your name and an H3 with "Web Designer," then save and refresh your browser.

2 Exercise 2.2 - Header Comment: Add a comment above your H1 that identifies it as the header of your webpage, and confirm it does not display in the browser.

Chapter 3: Elements, Paragraphs and Attributes

Elements

All HTML documents are made up of HTML elements. These start with a start tag and end with the closing tag. Some elements do not require a closing tag; these are known as single (or “void”) elements. An example of a single element is the break tag, written `<br>` .

To gather a clearer, logical understanding: HTML documents are just scripting, or branching, with elements to create a web page.

Paragraphs

To create a paragraph in HTML, you simply use the `<p>` element and close it with `</p>` . Type a paragraph about your interest in web design, and be sure to close your paragraph correctly. Save your code and refresh your browser. You should see your full paragraph alongside the other heading elements.

When typing paragraphs, think of writing an essay. When you create a new paragraph in your essay, you are creating a new element. So with each paragraph in your code, make sure to close out the old one and then create a new element. Let’s place a comment above our paragraph:

```
<!-- This is the start of the main information on the webpage. -->
<p>I enjoy teaching others how to code and create software applications!</p>
```

Attributes

Attributes are additional information about an element or tag, including modifiers. The value of an attribute is what modifies it. We will learn more about attributes and CSS in the CSS book. We can use attributes to change things in text or containers on our webpage, such as positioning, color, size and many more.

For example, we will center our H1 and H3 elements on the screen. In HTML, appearance is controlled with CSS, so we add a `style` attribute to the opening tag. The `style` attribute holds CSS declarations (property: value;). Try it!

```
<h1 style="text-align: center;">Your Name</h1>
<h3 style="text-align: center;">Web Designer</h3>
```

2026 Best Practice: Older tutorials used `align="center"` and `bgcolor`. Those attributes are obsolete in HTML5. Use CSS instead, ideally in an external stylesheet (you will move this inline style there in the CSS book).

Here is a table of other tags you can use to format text:

Prefer `<strong>` and `<em>` over `<b>` and `<i>` when the text carries meaning (importance or emphasis), because screen readers can announce that meaning. Use `<b>` and `<i>` only for purely visual styling.

Tag	Effect
<code></code>	Bold text
<code><small></code>	Smaller text
<code></code>	Important text
<code><del></code>	Deleted text
<code><i></code>	Italic text
<code><ins></code>	Inserted text
<code></code>	Emphasized text
<code><sub></code>	Subscript text
<code><mark></code>	Marked text
<code><sup></code>	Superscript text

Practical Exercises: Chapter 3

1 **Exercise 3.1 - Paragraph Practice:** Write a paragraph about your interest in web design inside a `<p>` element, with a comment above it.

2 **Exercise 3.2 - Center Alignment:** Center your H1 and H3 elements using the style attribute (`text-align: center;`).

3 **Exercise 3.3 - Text Formatting:** Use at least three tags from the table above in your paragraph.

Chapter 4: Images, Lists and Tables

Images

Adding images to your webpage will be essential as you progress as a developer. This is really fun to do! For the example in this book we will use an avatar generator to add an image to our portfolio. Head to an avatar maker website and create an avatar. Once you are finished, follow the steps below to increase your skill level by adding this image to your webpage.

When working with external documents and files in HTML, it is good practice to have relative directories, or folders, for each type. Go to your Portfolio folder and add a new folder named “images.” Then download or place your avatar image in that folder. The correct way to add an image to your HTML file is with the image element:

```

```

```

```

2026 Best Practice: Always write meaningful alt text. Set width and height so the page does not jump while the image loads, and add `loading="lazy"` to images far down the page. Use modern formats such as WebP or AVIF for smaller files.

There are some key things to note when using the image element. `img` tells the HTML document what to look for, in this case an image. The `src` attribute tells the HTML file where the source of the image is coming from; ours is “images/avatar.png.” The `alt` attribute gives the alternate text for the image in the event it doesn’t load properly. This also helps users who may have difficulty with sight, or who use features such as text-to-speech, understand what is being displayed.

Lists

In HTML, lists serve a variety of uses. You have ordered lists, initiated with the `<ol>` tag, as well as unordered lists, initiated with the `<ul>` tag. Both need list items, which we add with `<li>` for each item.

For your portfolio, let’s add an unordered list with 4 list items. This will later become our navigation menu when styling in CSS. Be sure to add a comment above the list element to let other developers know this is a navigation menu container. For the 4 item names, add: home, experience, services, contact.

```
<!-- This is the navigation menu container -->
<ul>
  <li>home</li>
  <li>experience</li>
  <li>services</li>
  <li>contact</li>
</ul>
```

Tables

Tables are used for many cases; sometimes you can use a table instead of a list and vice versa. For your portfolio we will not be adding a table, but it is great to know about them. In

HTML you create a table with `<table>` , then divide the table into rows, `<tr>` , then divide the rows into columns, called table data, `<td>` . You can add anything you want inside the table data section: images, text, links and more. A quick example:

```
<table>
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
  </tr>
  <tr>
    <td>Column 1</td>
    <td>Column 2</td>
  </tr>
</table>
```

This demonstrates a table with 2 rows and 2 columns. **Try it!**

Practical Exercises: Chapter 4

1 **Exercise 4.1 - Add Your Avatar:** Create an “images” folder, add your avatar, and display it with the image element and descriptive alt text.

2 **Exercise 4.2 - Navigation List:** Build the 4-item unordered list (home, experience, services, contact) with a comment above it.

3 **Exercise 4.3 - Try a Table:** Create a table with 2 rows and 2 columns and view it in your browser.

Chapter 5: Links, Block and Inline Elements

Links

Links are one of the most essential parts of your web page. You can add links to images or text that give your user the ability to click and be directed to another file, web page or location. We add links with the anchor tag and the href attribute, which holds the destination.

You can also add the target="_blank" attribute to the tag. It is best used for links that lead outside of your web page's directories, because it opens the link in a new tab. Simply place it after the closing quotation of your link's URL:

```
<a href="https://www.youtube.com" target="_blank" rel="noopener noreferrer">YouTube</a>
```

2026 Best Practice: When you open a link in a new tab, add rel="noopener noreferrer" and make the link text say where it goes. Only use target="_blank" when it genuinely helps the visitor.

Above, we created an unordered list. Let's turn each list item into a link. These links will lead nowhere for now, until we get the rest of the portfolio built. For the text in each list item, wrap the link tag around the text and not the list tag:

```
<ul>
  <li><a href="#">home</a></li>
  <li><a href="#">experience</a></li>
  <li><a href="#">services</a></li>
  <li><a href="#">contact</a></li>
</ul>
```

Block Elements and Div Containers

HTML block elements start a new "line" or section of information. Examples of block elements are <div> , <p> , <h1> – <h6> , and <table> .

For your portfolio we are going to transform some of the code we have built already by creating block elements and building div containers. The best thing you have done to make this possible is create comments in your code. We will use the <div> element quite frequently in web development and some forms of software engineering. It is a block element used as a container for other HTML elements, and it is greatly combined with CSS for styling and JavaScript for bringing life to our page.

Let's build a div container for our header, then break down its key aspects:

```
<div class="header" id="header">
  <h1>BioGlytch Studios LLC</h1>
  <h3>Web Designer</h3>
</div>
```

You have the opening div tag. Then the **class** name, which is what we use mainly for CSS styling and is identified in CSS as `.header` (note the period). Lastly, the **id** name, which is used for JavaScript and is identified in CSS as `#header` (note the hashtag). We will dive deeper into CSS and JavaScript in later lessons, but for now this is what you need to know to create styling sheets for your elements.

If you have been following along, you should have 3 comment sections. Let's create 4 div containers: **header, main, navigation, footer** . Move your navigation area above your main section.

What we have done is create 4 div containers, turning our multiple elements into children of their respective parent containers. For example, instead of having our name and career in the header section as 2 block elements that we style separately, when we place them in a header container we can style them at the same time with the same code, without interfering with other elements in other containers. Making our code manageable and easily readable is most important, and div elements increase the readability and organization of our webpage.

Modern HTML5 gives us **semantic elements** that describe their purpose. Search engines, screen readers and other developers understand them better than a pile of generic divs. Use them for the main parts of your page:

- `<header>` – the introductory area of a page or section.
- `<nav>` – a group of main navigation links.
- `<main>` – the main content of the page (use only one per page).
- `<section>` and `<article>` – a themed group of content, or a self-contained piece of content.
- `<aside>` – related side content.
- `<footer>` – the closing area, such as contact details and copyright.

Use `<div>` only when no semantic element fits, for example as a wrapper to style a group. A semantic version of your portfolio structure looks like this:

```
<header>
  <h1>Your Name</h1>
  <h3>Web Designer</h3>
</header>
<nav aria-label="Main">
  <ul>
    <li><a href="#home">home</a></li>
    <li><a href="#experience">experience</a></li>
  </ul>
</nav>
<main>
  <p>Welcome to my portfolio.</p>
</main>
<footer>
  <p>&copy; 2026 Your Name</p>
</footer>
```

Inline Elements

Inline elements are displayed without breaks in lines. Examples are `<a>` , `<b>` , `<i>` , `<span>` and `<img>` . Some inline elements will require CSS styling, while others, such as `<i>` for italics, are freestanding when it comes to their usage.

Practical Exercises: Chapter 5

1 **Exercise 5.1 - Linked Navigation:** Turn each navigation list item into a link by wrapping the anchor tag around the text.

2 **Exercise 5.2 - External Link:** Add a link to a website outside your folder that opens in a new tab using `target="_blank"`.

3 **Exercise 5.3 - Div Containers:** Create header, navigation, main and footer (use the semantic

elements or div containers), with the navigation placed above main.

Chapter 6: Forms

GET vs. POST

Collecting user information is very important in HTML. Capturing information with input has become increasingly easy over the past few years, with many contact form websites in use. It's always good to learn how to code and style your own forms for collecting information, especially if you plan on being a full stack developer.

We have 2 methods to use when collecting user information: GET and POST.

- **GET:** the form data will be visible in the page address.
- **POST:** best for updating data or when sensitive information is being captured. Security is better because the submitted data is not visible in the page address.

Input Types, Placeholder and Required

To have the data collected and sent to your server from the form, you need a way to take user input. We do this with the `<input>` element, which can take text, passwords, URLs, uploads and many more! Another key attribute in your form is `action`, which we use to redirect users after they submit a form.

For your project, we are going to add a very basic form to create a color wheel! Begin by creating the form block element, then add an input element with the type "color." Try out your color wheel!

```
<form action="/contact" method="post">
  <label for="color">Pick a color</label>
  <input type="color" id="color" name="color">
</form>
```

Every input needs a `<label>` linked with `for` and `id`, and a `name` so its value is sent. Labels make forms usable with screen readers and give touch users a larger tap target.

Other inputs are available, such as checkboxes, radio buttons, submit buttons and more! HTML5 added a new attribute for forms called **placeholder**. For input and textarea elements, placeholder offers a hint to let the user know what information to enter in that field. It is also best practice to use the **required** attribute on your input fields; simply add `required` at the end of your input element. This forces the user to enter information in order to continue.

```
<label for="name">Your name</label>
<input type="text" id="name" name="name" placeholder="Jane Smith" autocomplete="name" required>
<label for="email">Email</label>
<input type="email" id="email" name="email" autocomplete="email" required>
```

A placeholder is only a hint. It never replaces a label. Use the right type (email, tel, url, date) so phones show the correct keyboard and browsers validate for you, and

autocomplete so browsers can fill in details.

Form-Building Tools

Websites such as formsubmit.co and many others help create forms:

- **Google Forms:** A popular option for surveys, quizzes and forms. It's user-friendly and integrates well with other Google services.
- **Typeform:** Beautifully designed forms with a focus on a conversational interface, making form-filling more engaging.
- **JotForm:** A drag-and-drop interface with a variety of templates and customization options.
- **Formstack:** Form-building tools with easy integration with other applications.
- **Wufoo:** Known for its user-friendly interface and customization options for online forms.
- **Cognito Forms:** Flexible and powerful forms for a variety of uses, with customization and integration options.

Practical Exercises: Chapter 6

1 **Exercise 6.1 - Color Wheel:** Build a form containing an input of type "color" and test it in your browser.

2 **Exercise 6.2 - Placeholder and Required:** Add a text input with a placeholder hint and the required attribute, then try to submit it empty.

3 **Exercise 6.3 - GET or POST:** Explain in your own words when you would use GET and when you would use POST.

Chapter 7: Colors, Audio and Video

HTML Colors

HTML colors are written with hexadecimal characters: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. We use the wheel of red, green and blue light (RGB) when determining our colors. Hex values are written with the hashtag symbol (#), followed by 3 or 6 hex characters. This is supported by all browsers, and there are over 16 million color combinations using hex values.

Go to your body tag and add a style attribute with the background-color property. We are going to **change the color of our site!** Go to your color wheel and pick a color that will show the black text on your site well.

```
<body style="background-color: #ADD8E6;">
```

2026 Best Practice: bgcolor is obsolete. Use the CSS background-color property. When you choose colors, check that text and background have enough contrast (at least 4.5:1 for normal text under WCAG 2.2).

Audio and Video Elements

HTML5 introduced the <audio> and <video> elements, which enable us as developers to embed audio and video in our web pages. The element creates a player inside the browser, and the text inside the element will display if there is a failure with your media files. There are two ways to specify the source URL for the file; we can use the source attribute, similar to how we find images.

Attributes of the audio (or video) element include:

- **controls:** displays audio controls such as the play/pause button.
- **preload:** hints how much to load up front (none, metadata or auto).
- **muted:** starts silent. Browsers block autoplay with sound, and autoplay is hard on many users, so avoid autoplay unless the video is muted and decorative.
- **loop:** replays the file every time it finishes.

There are 3 supported file formats for audio elements: MP3, WAV and OGG.

```
<video controls width="640" poster="images/poster.jpg">  
  <source src="video/intro.webm" type="video/webm">  
  <source src="video/intro.mp4" type="video/mp4">  
  <track kind="captions" src="video/intro.en.vtt" srclang="en" label="English">  
  Your browser does not support the video element.  
</video>
```

Always provide captions (a <track> element) or a transcript for audio and video so everyone can use your content.

```
<audio controls>
  <source src="audio/song.mp3" type="audio/mpeg">
  Audio Element Unable to Initiate.
</audio>
```

Practical Exercises: Chapter 7

1 **Exercise 7.1 - Background Color:** Add a background-color style to your body tag using a color that keeps your black text readable.

2 **Exercise 7.2 - Audio Player:** Embed an MP3 file with the controls attribute and fallback text, then test the player.

2026 Best Practices for HTML

Use this checklist on every page you build:

- **Start every page correctly.** `<!DOCTYPE html>`, `<html lang="en">`, `<meta charset="UTF-8">`, the viewport meta tag and a unique `<title>`.
- **Use semantic elements.** header, nav, main, section, article, aside and footer instead of div soup. Use one `<h1>` per page and keep headings in order.
- **Keep content, style and behavior separate.** HTML for structure, CSS in an external file for design, JavaScript in its own file.
- **Skip obsolete elements and attributes.** Do not use `<center>`, `<font>`, `align`, `bgcolor`, `<marquee>` or `<applet>`.
- **Make it accessible.** Meaningful alt text, labels on form fields, enough color contrast, keyboard-friendly links and buttons, and captions for video.
- **Build mobile-first.** Test on a phone-sized screen; a page should never need sideways scrolling.
- **Use native features.** `<details>` for expandable content, `<dialog>` for pop-ups, `<button>` for actions and `<a>` for links.
- **Keep pages fast.** Compress images, give them width and height, and lazy-load below-the-fold media.
- **Use HTTPS and protect privacy.** Never put passwords or private data in URLs, and only collect the information you need.
- **Validate and test.** Run your page through the W3C Markup Validator and test in at least two browsers.

Conclusion

Recap of Key Concepts

Congratulations on completing this book! Now that you have the fundamentals of using HTML, we highly recommend using the CSS book to deepen your knowledge and skills within web development. Below are some of the things we covered that will help aid your career in software engineering.

- **Headings & Comments:** HTML includes six levels of headings ranked from 1 to 6. Properly open and close your tags, and use comments to enhance code understanding.
- **Elements:** All HTML documents consist of elements with an opening and closing tag. You learned about block and inline elements, paragraphs, attributes, images, lists, tables, links and more.
- **Forms:** Collecting user information is crucial. You learned about form elements, input types and attributes, the GET and POST methods, and the action attribute.
- **Colors, Audio & Video:** You explored hexadecimal colors, changed your background color, and embedded audio and video using controls, loop and autoplay.

Encouragement to Explore and Experiment

As you conclude this book, you've taken significant strides toward becoming a proficient coder. Your journey doesn't end here – it's just the beginning. Continue to explore, experiment and, most importantly, enjoy the process of bringing your ideas to life through code.

- **Stay Curious:** Explore new tools, tags and emerging web technologies.
- **Practice Regularly:** Keep building and refining your portfolio page.
- **Embrace Challenges:** Fixing bugs builds deep technical confidence.

HAPPY CODING!

Appendix

Comprehensive Glossary - Section A (Attributes, Elements & Page Structure)

Alt Text: The alt attribute on an image that describes it when it cannot load, and for screen readers and text-to-speech users.

Anchor (<a>): The element used to create a link to another page, file or location.

Attribute: Additional information about an element that modifies it, such as align, src, class or href.

Autoplay: An audio/video attribute that starts playback as soon as the file is ready, without user permission.

Block Element: An element that starts a new line or section of information, such as div, p, h1 or ul.

Body (<body>): The part of an HTML document that holds everything displayed on the page.

Browser: Software that reads HTML documents and displays them as web pages, such as Chrome, Firefox, Edge or Safari.

Bug: An error or inconsistency in code that causes a page not to work as intended.

Child Element: An element nested inside another element, called its parent.

Class: An attribute used mainly for CSS styling; written as .name in CSS.

Closing Tag: The tag that ends an element, written with a slash, such as </p>.

Comment: Text in code that the browser ignores, written to help developers understand the code.

Controls: An audio/video attribute that displays playback controls such as play and pause.

CSS: Cascading Style Sheets; the language used to style the look and layout of HTML.

Debugging: Finding and fixing bugs in code.

Div (<div>): A block element used as a container for other HTML elements.

DOCTYPE: The declaration at the top of an HTML file that tells the browser it is HTML5.

Element: A part of an HTML document, usually made of an opening tag, content and a closing tag.

Extension: An add-on that gives a program new features, such as Live Server in Visual Studio Code.

File Path: The location of a file, such as images/avatar.png.

Form: An element used to collect information from users and send it to a server.

GET: A form method that shows the submitted data in the page address.

Head (<head>): The part of an HTML document holding information about the page, such as its title.

Comprehensive Glossary - Section B (Forms, Publishing & Web Basics)

Heading: One of six levels of title text, from h1 (largest) to h6 (smallest).

Hexadecimal: A base-16 system (0-9 and A-F) used to write HTML color values.

Host / Hosting: A service that stores your website files on a web server so others can visit them.

HTML: HyperText Markup Language; the markup language used to structure web pages.

HTTP / HTTPS: The protocols browsers use to request web pages; HTTPS adds encryption for security.

Hyperlink: Text or an image a user can click to go to another file, page or location.

ID: An attribute that uniquely names an element; used by JavaScript and written as #name in CSS.

Inline Element: An element displayed without breaking the line, such as a, b, i, span or img.

Input (<input>): A form element that takes user data such as text, passwords, colors and uploads.

IP Address: A unique numeric address assigned to a device connected to a network.

JavaScript: The programming language that makes web pages interactive.

List Item (): A single item inside an ordered or unordered list.

Live Server: A Visual Studio Code extension that previews your page in the browser and refreshes it as you save.

Markup: The tags and elements written in HTML to structure content.

Navigation Menu: A group of links that helps visitors move around a website.

Opening Tag: The tag that begins an element, such as <p>.

Placeholder: A form attribute that shows a hint inside an input field.

POST: A form method that keeps submitted data out of the page address; best for sensitive information.

Portfolio: A website showcasing your work, skills and experience.

Relative Path: A file location written relative to the current file or folder.

Required: A form attribute that forces the user to fill in a field before submitting.

RGB: Red, Green, Blue; the light colors combined to create screen colors.

Server: A computer that stores and delivers website files to browsers.

Single (Void) Element: An element with no closing tag, such as `<br>` or `<img>`.

Source (src): The attribute that tells an element where its file comes from.

Table: An element that organizes data into rows (tr) and columns (td).

Tag: The code in angle brackets that marks the start or end of an element.

Text Editor: Software used to write code, such as Visual Studio Code.

URL: Uniform Resource Locator; the web address of a page or file.

Web Server: A computer that stores web page files and delivers them to browsers.

WWW: World Wide Web; the system of linked pages accessed through the internet.

Useful Websites and Tools

This section lists tools, websites and communities that provide valuable resources, support and practice for continued learning in web design.

Tools for Writing and Testing Code

Code Editors and Extensions

- Visual Studio Code: code.visualstudio.com
- Live Server extension by Ritwick Dey (Extensions tab in Visual Studio Code)
- Prettier (code formatter) and Auto Rename Tag extensions for Visual Studio Code

Testing and Debugging

- W3C Markup Validator: validator.w3.org
- Browser Developer Tools: press F12 in Chrome, Edge or Firefox
- Can I Use: caniuse.com (check browser support for HTML features)

Publishing and Hosting

- GitHub Pages: pages.github.com
- Netlify: www.netlify.com
- Cloudflare Pages: pages.cloudflare.com

Design Resources

Colors, Fonts and Images

- HTML Color Picker: www.w3schools.com/colors/colors_picker.asp
- Coolors (palette generator): coolors.co
- Google Fonts: fonts.google.com
- Unsplash (free photos): unsplash.com

Forms

- FormSubmit: formsubmit.co
- Google Forms: forms.google.com
- Typeform: www.typeform.com
- JotForm: www.jotform.com

Learning and Reference

Documentation and Tutorials

- Mozilla Developer Network (MDN): developer.mozilla.org
- W3Schools: www.w3schools.com
- web.dev: web.dev/learn
- freeCodeCamp: www.freecodecamp.org
- Khan Academy: www.khanacademy.org

Online Courses

- Coursera: www.coursera.org
- edX: www.edx.org
- The Odin Project: www.theodinproject.com

Books

- "HTML and CSS: Design and Build Websites" by Jon Duckett
- "Learning Web Design" by Jennifer Niederst Robbins
- "Eloquent JavaScript" by Marijn Haverbeke
- "The Pragmatic Programmer" by Andrew Hunt and David Thomas

Technical Support and Communities

Technical Support

- BioGlytch Studios LLC Support: software@bioglytchstudios.tech | <https://bioglytchstudios.tech> | (989) 900-6747
- Gloria Coles Flint Public Library: ask a librarian about computer access and technology classes
- Microsoft Support: support.microsoft.com
- Apple Support: support.apple.com

Programming Communities

- Stack Overflow: stackoverflow.com
- GitHub: github.com
- Reddit: [r/learnprogramming](https://www.reddit.com/r/learnprogramming) and [r/webdev](https://www.reddit.com/r/webdev)
- Tech Support Guy: www.techguy.org