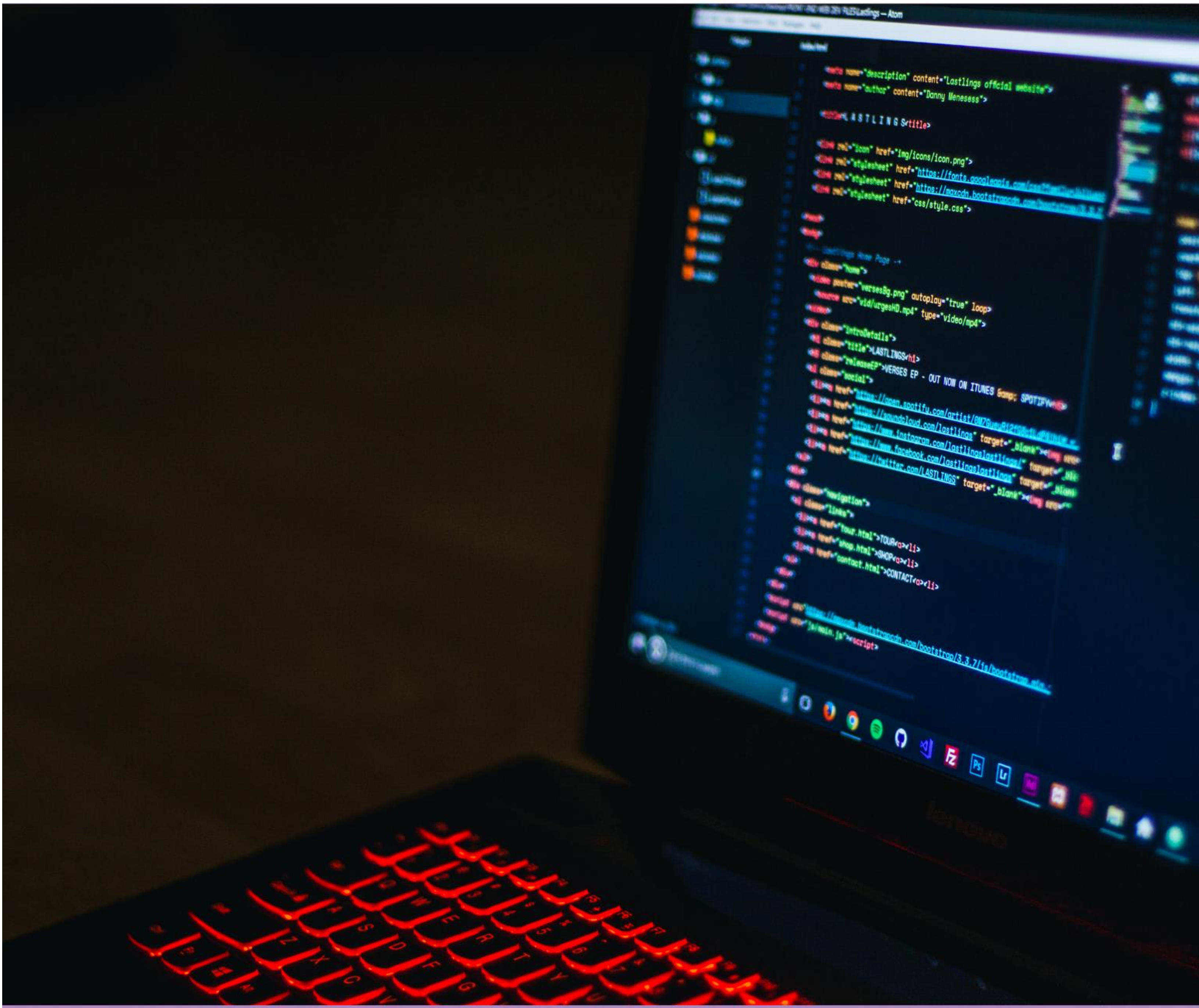
Computer Coding

Learn to Program



Web Development

Astro Clare Technology

Astro Clare Technology © Flint, MI EST 2022

" HELLO CODER "

We at Astro Clare Technology are proud to have you receive this book! There will be many fun activities for you to enjoy. If you are a parent purchasing this book for your child to venture into they great world of becoming an software engineer, we thank you for trusting us ! We have many great activities for you and your child to enjoy together.

Our main objective is to get individuals to begin their journey in coding, programming and computing technologies. We see the lack of resources decline over the years and understand that purchasing these courses at Universities are even more challenging financially and timely. So we will due more than just "teach" you how to become the greatest engineer!

You are going to learn many skills such as: critical thinking, logical outlooks, problem solving skills, discipline and career ready structure and even more !

Each successful assessment from you will grant an Certificate from Astro Clare Technology along with an PDF of your course completion to show your next employer.

Once again, we thank you for trusting in Astro Clare Technology and hope you put your best code forward.

Clarence Scott
CEO & Founder

Overview

Web Development is the process of creating websites, webpages, or web applications that are accessed via the internet. It involves various aspects, including designing, coding, debugging/testing, and maintenance to ensure proper functionality and user experience.

Web Development is an overall dynamic and evolving field that requires continuous learning and adaption to new technologies and best practices. Collaboration and communication skills are also very important, as developers often work in teams with designers, stakeholders and content creators to deliver successful web projects.

Some Key Components and technologies involved in web development:

Pre-Requisites

In order to begin coding with us you only need 3 things: Computer (with monitor), Text Editor and the drive to want to code.

First, let’s make sure you have the visual studio code software downloaded. Head over to Microsoft’s website and download Visual Studio Code, or click the link here Visual Studio Download. In the Extensions tab after you have it downloaded install the **extension Live Server by Ritwick Day** and you will be ready to continue. If you need help refer to our Youtube Video **“ Setting Up Visual Studio for Development”.**

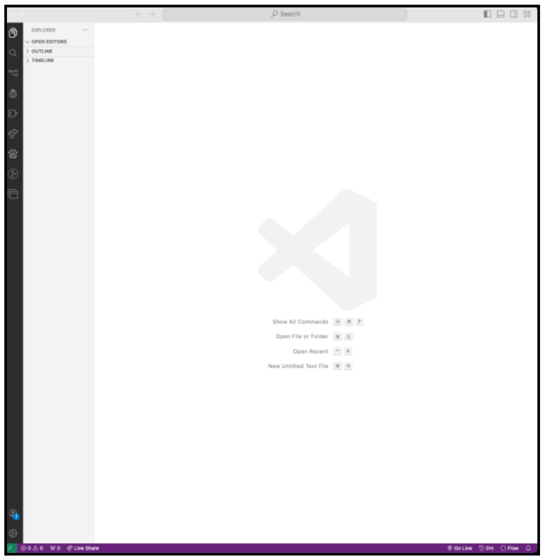
Secondly, ensure you are in a quiet setting and can focus on coding and studying the information in this book. Focus is key, determination is power. Code is the goal.

Lastly, grab a snack and something to drink (always have a water near you) so that you can stay mentally strong while studying and coding. The brain will become very tired and you will become mentally drained if you do not.

Creating Our Docs

"Hello World"

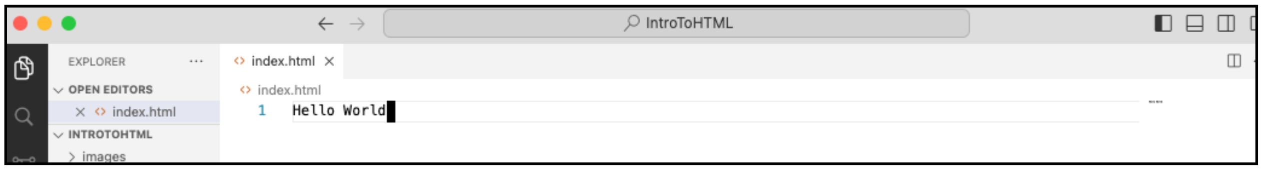
Now that you are ready to code, let's take a look at our setup in visual studio.
If everything was done correctly your screen should look like this:



If so, you have successfully downloaded Visual Studio Code. If you hover over the "Open Editors" section you can add a new file. Do that.

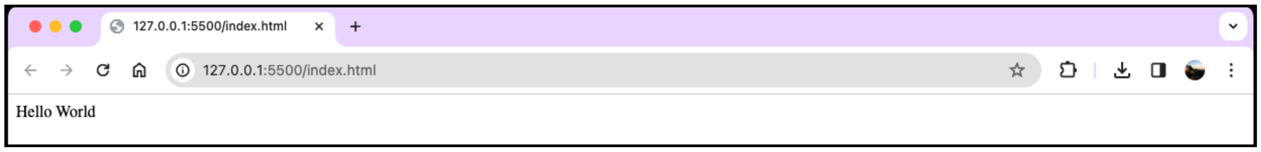
In that file we are going to add the text: "Hello World" and save the file. Once the popup shows to name your file, create a folder where you are at and name the folder "Portfolio" and then name your text file "index.html" and save it.

Reopen Visual Studio with this new folder by either closing the app and opening in the folder path by right-clicking or by clicking the "file" tab in the Visual Studio window and proceeding from there.



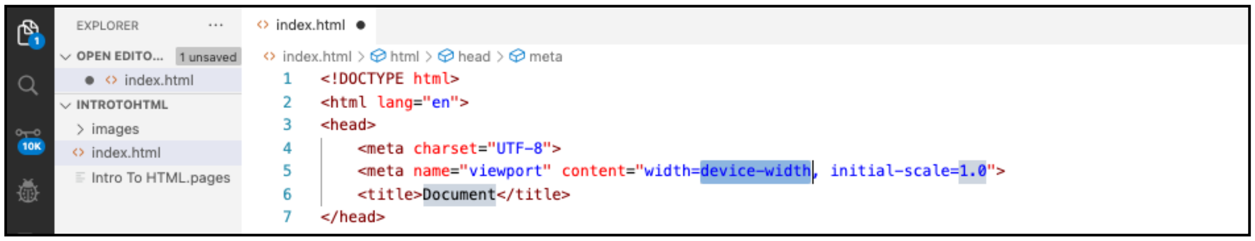
```
Hello World
```

Once done your new layout should look like this.
If you have a similar layout the bravo you are ready! Assuming you have downloaded the Live Server extension down in the right hand corner you should see "Go Live". If so click that and watch what happens next.



Hello World

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 click enter and your new Visual Studio page should look like this:



```
<!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>
```

Table Of Contents

c. Construct and analyze styles that format backgrounds and borders
    i. Border-color, border-style, border-width, background properties, colors
d. Construct and analyze styles that create a simple responsive layout
    i. Units of measurement (percentage, pixel, emphasized text (em), view-width, view-height), viewport and media query, frameworks and templates, working with breakpoints, grids

VI. Accessibility, readability, and testing
  a. Construct well formed HTML and CSS markup to industry best practices
    i. Reusing rules and rule sets, commenting, web safe fonts, cross platform usability, separation of HTML & CSS
  b. Apply Accessibility principles and evaluate content accessibility
    i. Text alternatives, color contrast and usage, legibility of typography, tab order, text resizing, text hierarchy, translate
  c. Evaluate the structure of integrity of HTML and CSS
    i. Syntax errors, tag mismatch, cascading issues.

HTML Fundamentals

Markup with metadata elements

When you are working with HTML elements, they are essential for ensuring the proper management, organization and increased value of your webpage. They are effective for receiving information, discovery, interpretation, and preservation in web applications. We are going to cover a few in detail. Script, no script, style, link, meta tags (encoding, keywords, viewport, and description).

- Script: Used to embed or refer to executable code within an HTML document. JavaScript code can be included within <script> tags, rather than external document.
- No Script: This element provides alternate content to users who have disabled scripts in their browsers or whose browsers don't support scripting languages.
- Style: <style> tags are used to initiate CSS (Cascading Style Sheets) within an HTML document. This allows you to control the presentation and layout of your HTML content.
- Link: The <link> element is used for external resources, such as stylesheets, icons, JavaScript, or other documents. You would use <link> when linking access to your CSS stylesheet through your HTML document.
- Meta Tags: These are used to provide metadata about the HTML Document
    - Encoding: <meta charset=" UTF-8" > specifies the character encoding for your document.
    - Keywords: <meta name=" keywords" content=" word1, word2..." > Provides keywords relevant to the documents content.
    - Viewport: <meta name=" viewport" content=" width=device-width, initial-scale=1.0"> sets the viewport properties, mainly used for responsive design.
    - Description: <meta name=" description" content=" Brief description of your webpage"> provides a summary of the page's content. (30-150 words)

Construct well-formed page markup

When creating well-formed page markup, you should follow certain conventions to have proper HTML documentation layout. Following these guidelines will help give you proper structure and contain all the data needed for search engines and browsers to interpret your webpage correctly.

- DocType Declaration: This is the very first line of all HTML documents. It tells the browser which version of HTML the page is written in. <!DOCTYPE html> is the declaration for HTML5, which is what you should be using.
- Html: the <html> element is the root element of an HTML page. All other elements should always be contained in it.
- Head: the <head> element contains meta-information about the documents. (see meta-tags above)

- Body: the <body> element contains the content of your HTML document that will be shown in the browser.
- Proper syntax: HTML syntax must be correct for the document to render. This includes using opening and closing tags correctly, nesting elements properly, as well as using attribute values properly.
- Closing tags: Every opening tag must have a corresponding closing tag, apart from self-closing tags like <img> or <br>
- Common Symbols: Some characters have special meaning in HTML and must be represented using character entities. ‘&lt;’ represents the less-than sign ‘<’. Or using the ampersand ‘&’ should be represented using ‘&amp;’. These are used to avoid parsing errors to display the characters correctly in the browser.

CSS Fundamentals

Analyze and Implement inline styles, embedded style sheets and external stylesheets

Inline styles are CSS styles directly applied to individual HTML elements using the 'style' attribute. An attribute is a CSS property that is used to assign a style or behavior to an HTML element. Inline styles are useful for changes or applying unique styles to specific elements.

```
<p style="color: blue; font-size: 18px;">This is an inline example.</p>
```

Embedded style sheets are CSS styles that are defined within the <style> element in the head section of an HTML document.

```
<head>
    <title>Document</title>
    <style>
        p{
            color: blue;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <p>This is a paragraph.</p>
</body>
```

External Stylesheets are CSS files that are linked to an HTML document using the <link> element in the <head> section. External stylesheets allow you use your CSS styles outside of your HTML document. Making your code easier to maintain, while applying consistent styles across multiple HTML pages.

```
<head>
    <title>Document</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<p>This is paragraph. I am being styled externally!</p>

</body>
```

There is a time and place for using these different stylesheets. Inline styles are for quick styling changes or unique styles on specific elements. Embedded styles are used to apply styling to multiple elements in a single document. External sheets are used to apply consistent styles across multiple HTML pages or when you want to keep your CSS styles separate from your HTML. This would be for better organization and maintenance.

Construct and Analyze rule sets

Valid syntax for CSS rule sets consists of a selector and one or more declarations that are enclosed in curly braces ‘{}’. By constructing and analyzing CSS rule sets, you can effectively style your HTML document to achieve the desired visual presentation. Understanding selectors and valid syntax is crucial for writing efficient CSS code and maintaining it.

```
p{
    color: blue;
    font-size: 18px;
}
```

In this example ‘p’ is the selector, which selects all **‘<p>’** elements. The declarations, **‘color:blue;’** and **‘font-size:16px;’,** define the styles to be applied to the paragraph elements.

As you continue to use selectors it is important to be aware of the 4 selectors you will be using: class selector, id selector, element selector, pseudo-selector.

Class selectors begin with a period symbol (.) followed by the class name of your choosing. For example, ‘**.this-class**’. Id selector begins with the hash symbol (#) followed by the id name of your choosing. For example, ‘**#my-id**’. Element selectors, selects elements based on their tag name. For example, the **p** element selects all **<p>** elements. Pseudo-selectors allow you to style elements based on their state or position in the document. For example, we can use ‘**:hover**’ to select an element when the mouse cursor is over it.

Document Structure using HTML

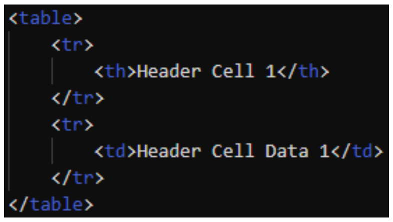
Construct and Analyze markup to structure content and organize data

Your HTML document is made up of structured data, semantic elements, appropriate navigation, and form attributes. These all help to maintain proper structure to your HTML content and data. Data organization consists of tables, headings, paragraphs, break points, div containers, ul and many more. Semantic elements include headers, sections, footers, captions and many more. Navigation elements include anchors, folder hierarchies, area, links etc. Form elements include attributes, action, methods, input types, textarea, labels and many more. By effectively understanding and using these elements, you will be able to create well-structured and meaningful documents,

Markup using data structures

Tables are used to organize data into rows and columns.

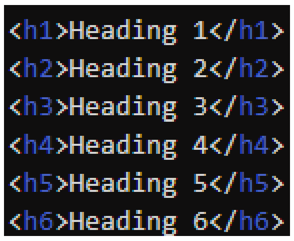
1. <table> – defines a table
2. <tr> - defines a table row
3. <th> - defines a table header cell
4. <td> - defines a table data cell



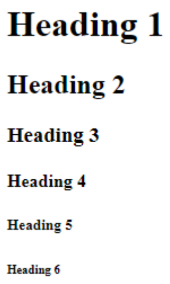
```
<table>
    <tr>
        <th>Header Cell 1</th>
    </tr>
    <tr>
        <td>Header Cell Data 1</td>
    </tr>
</table>
```

Proper layout of a table, above.

Headings are used to define headings of different levels of importance, where h1 is the highest level and h6 is the lowest level. **(h1-h6)**



```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```



Paragraphs are used to insert paragraphs with the <p> element.

Line break <br> inserts a line break to create a new line or row. Can be nested in <p> elements to create a new row in your paragraph.

Horizontal rule <hr> inserts a horizontal line break or thematic break.

Div <div> is used to create a new section or division in an html document.

Span <span> defines a section of text with no specific meaning.

Lists are defined in 2 different ways but the items are inititaed the same. **Unordered** bulleted lists, are defined using **<ul>**, with the items being inserted using **<li>**. **Ordered** numbered lists are defined using **<ol>**, with the items being inserted using **<li>**.

Markup using semantic elements

Semantic elements in HTML provide meaning to the content they wrap. This makes for search engines and developers to understand the structure of the webpage. This includes **<header>** which defines the header for a document or section. Navigation **<nav>** is used to define navigation links, **<section>** is used to define a section in a document. Independent pieces of content are defined with the **<article>** element. **<aside>** defines content placed aside from the content it is placed in. Footer is defined with **<footer>** to express the footer for a document or section. To provide additional details to view or hide, **<details>,** would be used. To provide a visible heading for a <details> element you can use **<summary>.** To define self-contained content, images or diagrams for exmaple, would be defined with **<figure>.** You can provide a caption for <figure> elements with **<caption>.**

Markup using elements that implement navigation

Navigation elements are used to navigate within or outside of your webpage. This could be to external links or links within your HTML folder heirarchy map.

Target and **anchor** links are used to create hyperlinks **<a href="url"> Link Text</a>**

```
<a href="http://youtube.com">YouTube</a>
```

Bookmark anchors used for creating internal links within the same document. These are anchors with a name attribute

```
<header id="top">
 <p>Some Content</p>
</header>
<div class="section1">
    <p>Some more Content</p></div>
<div class="section2">
    <p>Some more Content</p></div>
<div class="section3">
    <p>Some more Content</p></div>
<a href="#top">Go to the top of the page.</a>
```

Relative and **Absolute** links are used to utilize different types of URLs. **Relative** links specify the path to a resource in the current document, only used to point to a file or a file path. **Absolute** links specify a full URL that includes a domain (complete address to a file or resource).

Absolute Domain Examples

- http://yourdomain.example/images/example.png
- //yourdomain.example/images/example.png

Relative Domain Examples:

- /images/example.png
- images/example.png

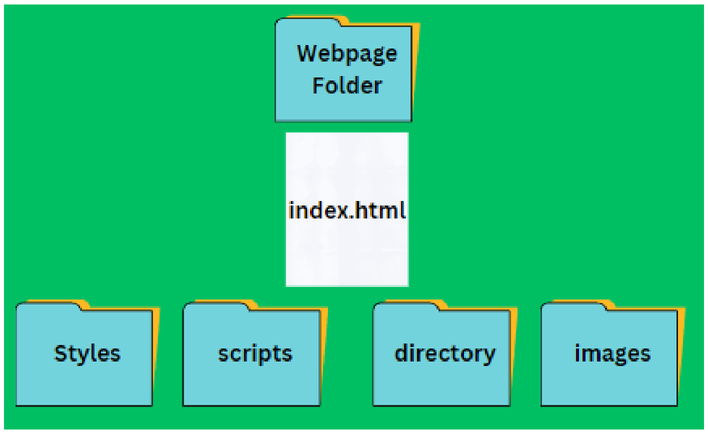
Navigation is used to construct linkage between pages or sections within a website.
Within a page to a different section:

```
<header id="top"></header>
<a href="#top">Connects to the header.</a>
```

Within a site to a different page:

```
<a href="sample.html">Sample Page Link</a>
```

It is extremely necessary to understand the organization of files and folders in a website **directory structure**, also known as your **hierarchy** map. All webpages, websites, webapps and software use a hierarchy map or directory structure. The image below shows the flow of a simple structure for an HTML website.



Area is used with the <map> element to define clickable areas in an image map. Including the coordinates for the specified clickable area is how you specify the clickable area.

```
<img src="coffee.jpg" alt="" usemap="#coffeemap">
<map name="coffeemap">
    <area shape="rect" coords="34,44,270,350" href="beans.jpg" alt="Coffee Beans">
</map>
```

Markup using form elements to build Forms
Form elements are used to build various forms for contact pages, checkout information when developing online stores or shops to purchase items and many more things. Let's cover some elements and build a sample form. We'll be using form attributes, which are actions that determine how form data will be submitted, using action and method.

Submission methods are used to send data from your HTML document. **Action** specifies the URL where the form data will be submitted. Websites like https://formsubmit.co/ have easy setup features that allow you to link your current email address with the **action** attribute to

receive emails from your website securely. **METHOD** relates to the HTTP method used to submit our form data **(GET or POST).**
GET vs. **POST**

- **GET** requests are used to request data from a specific resource and are sent in the URL of the request.
    - Can be cashed, have restrictions in data length, only used to request data, can be bookmarked, remains in the browser history, SHOULD NEVER be used when dealing with sensitive data.
- **POST** requests are used to send data to a server to create/update a resource, these are stored and sent in the body of the HTTP request.
    - Never cached, does not remain in the browser history, cannot be bookmarked, should be used when dealing with sensitive data, have no restrictions on data length.

There are multiple input types you will use in your forms. These include text, password, email, and phone input attributes and many more. There are optional attributes for validation and restrictions. Having a menu option in your form is essential for allowing multiple options to be selected from using **<select>** to define the dropdown list, and then **<option>** to define the options within the dropdown list. Creating an area for user to input text in your form is essential to taking in information using **<textarea>.** Labels are an element needed to label our input types, we can give each input type an label by adding **<label>** before each input.

```
    <label for="first-name">
        First Name
        <input type="text" name="firstname" id="fname" placeholder="I am name
input">
    </label>
```

Lastly, you want to allow your user to submit the form information. You can do this by adding a button element using **<button>** and apply multiple attributes to your button. There are 4 options to choose from: **b: button, d: disabled, r: reset, s: submit**.

Understanding and effectively using HTML form elements will help to create well-structured and semantically designed web applications, websites and webpages. Be sure to properly implement navigation and handle user input appropriately in your forms.

Multimedia presentation using HTML

Constructing HTML markup the introduces images

As a web developer or designer, images are a beautiful element that will bring life to any project. They are simple and fun to add, and easy to style with CSS. We embed images within the HTML document using the image element **<img>**. There are 7 attributes we mainly use for image elements: src, alt, width, height, title, srcset, sizes.

The **src** attribute specifies the URL (or pathway of the image file), **alt** provides the text for the image which is displayed if the image cannot load or for the use of screen readers for accessibility features. Using the **srcset** attribute allows you to specify multiple image files for different resolution or display sizes. We can change the size of the image with the **width** (adjusts the width properties of the image) and **height** (adjusts the height properties of the image) attributes, as well as the sizes attribute. The sizes attribute specifies the size of an image based on the different viewport widths. The typical layout for an HTML image markup element is as follows:

```
<img src="yourphoto.png" alt="Photo Description" width="500" height="600">
```

As a developer you can specify multiple sources for an image based on different criteria such as: device resolution or viewport size, with the **picture** element **<picture>**. Inside the picture element you can use the **<source>** element to specify different image sources. These source elements can use the srcset and sizes attributes to explore multiple image options.

Constructing HTML markup to analyze audio and video elements

The audio **<audio>** and video **<video>** elements are used to embed video and audio elements into your HTML document. These elements use most of the same attributes to find the path for the content and control the initialization of the content.

Video element uses the **src** attribute to specify the URL of the video. In addition to **src**, you can add the **controls** attribute to add playback options to the video player (play, pause, volume, etc.). **Autoplay** tells the video to automatically play when the page loads, adding the **loop** attribute causes the video to continuously play. The **poster** attribute allows you to choose an image to show while the video is downloading or until the user starts to play the video file.

```
<video src="yourpath/video"" autoplay controls loop
poster="yourpath/toimage"></video>
```

Audio element uses the same attributes as video apart from poster, **src** attribute to specify the URL of the video. In addition to src, you can add the **controls** attribute to add playback options to the video player (play, pause, volume, etc.). **Autoplay** tells the video to automatically play when the page loads, adding the **loop** attribute causes the video to continuously play.

```
<audio src="yourpath/audio" controls autoplay loop></audio>
```

HTML includes a track element **<track>** used to specify text for audio or video elements, that creates subtitles or captions. This has 4 attributes: **kind**: specifying the kind of track being used (subtitles, captions, descriptions); **src**: the URL of the text track file; **srclang**: specifies the language of the text track; **label**: specifies a label for the text track.

Source elements are used to specify alternative media resources for the **<video>** and **<audio>** elements. This can use the attributes: **src** for the URL of the media file, **type** specifies the MIME type of the media file (video/mp4, audio/mpeg).

In the event you need to embed another HTML page or external content within the current document you are working on, you can use the iframe element **<iframe>.** This element has 3 main attributes to use, **src**: specifies the URL of the embedded content, as well as **width** and **height** attributes to change the sizing of the iframe.

The proper utilizing of these elements will allow you to embed images, audio, and video content into your HTML documents effectively, to enhance your media presentation of your webpages.

Webpage Styling using Cascading Stylesheets (CSS)

When styling in CSS, constructing styles that have focus on positioning, formatting, backgrounds, borders, and responsiveness are what create a positive user experience and user interface. Let's take a dive into some of the properties of these focus areas.

Cascading style sheets have several **positioning** properties: **float**, **relative**, **absolute**, **static**, & **fixed**.

To allow an element to float on the left or right of its containing element we use the **float** property. While the **relative** property positions an element to its normal position, the **absolute** property positions an element to its closest positioned parent element. The **static** property sets the element to the default positioning for all elements. Lastly the **fixed** position makes an element positioned relative to the browser window.

The **overflow** property specifies what happens if the content of an element exceeds the allotted space.

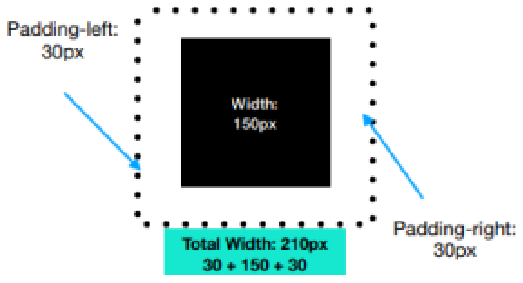
Height and **width** properties set the size of elements. Using the **max-width** or **max-height** properties allows us to set the maximum width or height of an element.

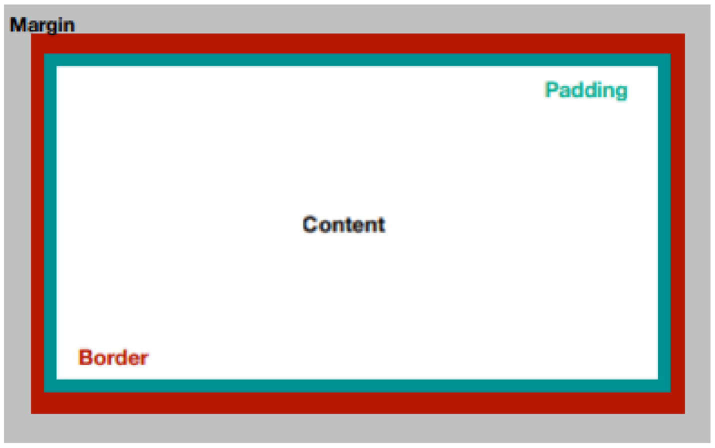
Properties for alignment are useful when organizing elements and sections. The **align** property can be used to position text or elements to the left, center or right of their containers. In addition to alignment properties, the **display** property is more used when displaying the alignment behavior of an element. 4 common attributes of display are: **block**, **inline**, **inline-block**, **flex**.

Using the **visibility** property determines if we allow an element to be visible or hidden.

Box model

Having a common understanding of the box model is essential when working with containers and elements whose styling will be affected by **content**, **padding**, **border**, and **margin** areas. **Margin** controls the space outside of an element, while **padding** controls the space within the element. **Border** is the space between your margin area and padding area., while the **content** is what is inside of the padding area.





When styling in CSS, constructing styles that have focus on text formatting creates easy to read content and exaggerates the user experience and user interface. Let's take a dive into some of the formatting properties of text.

When working with text formatting there are 3 things to think about: font properties, link styles and text formatting.

Font properties change several things in our text for HTML. You may want to change your text font in your HTML document, maybe to courier new & Calibri format. The **font-family** property will allow you to do that. Sometimes changing the color of your web sites text is needed to create easy to read content or a cool effect to your page. Adding the **color** property allows you to do that. Adjusting our **font-style** to create text that is italic, normal or oblique may be necessary as well to stress important text. Changing the size of our text can be done with pixels (px), inches (in), percentage (%) and many other size styling attributes using the **font-size** property. Adjusting the boldness of our text is simplified with the **font-weight** property giving us the options of (normal, bold, lighter, bolder). If text is needed to be displayed in small caps **font-variant** is the appropriate property to use for this attribute.

Link styles *and* ***text formatting*** work hand in hand. Changing the text **color** of your links requires the anchor tag to be included in the selector portion of your CSS.

Link styling example: ul li **a** {**color**: **green**;}

Text Formatting properties like **text-transform**, **text-shadow** and **text-overflow** are commonly used. Changing text to **uppercase**, **lowercase** or **capitalizing** the begging of each word in an element should be done with the **text-transform** property. Creating a mirror effect, neon effect or adding shadow to text and many more options should be done with the **text-shadow** property. When you have text overflowing in its inline direction the **text-overflow** property allows text to be clipped for a partial render or replaced by an ellipsis character.

When styling in CSS, constructing styles that have focus on backgrounds and borders creates added unique style to the user experience and user interface. Let's take a dive into some of the background and border properties of text.

Background properties can set and change the color, image, size and if we want the image to repeat. Changing the color of backgrounds is a common property to use in HTML ***background-color*** can have the attribute that specifies the name of the color. Example **rosybrown**. In addition, you could also use **RGB** format, for example **RGB (188, 143, 143).** The **HEX** format is also commonly used **#bc8f8f**. Lastly, **HSL** is another property to use that is represented by percentages, example **HSL (0, 25%, 65%).**

To change the background of your body element or another element or section, you will use the ***background-image*** property with the **URL (' ')** attribute. The URL attribute should include the file path to the image wanting to be used, for example **background-image: URL('filepath/image.png').** There are property attributes that can be added to the background-image property that allows the changing of the image itself.

Two background-image properties we will cover are background-size and background-repeat:

Background Size: **`background-size: auto|`*`length`*`|cover|contain|initial|inherit;`**	
auto	The background image is displayed in its original size.
length	Set the width and height of the image.
percentage	Sets the width and height as in percentage. The first value is width the second is height.
cover	Makes the image cover the entire container. This will cause the image to stretch or cut of edges.
contain	Makes the image fully visible.
Initial	Sets the property to its default value.
inherit	Takes the property from its parent element.

Background Repeat: **`background-repeat: repeat|repeat-x|repeat-y|no-repeat|initial|inherit;`**	
Repeat	This is the default, the image is repeated horizontally and vertically, but the last image will be clipped if it does not fit.
Repeat-x & Repeat-y	Repeat-x, repeats the image horizontally. Repeat-y, repeats the image vertically.
No-repeat	Background image is not repeated, it will only be shown once.
Space	Repeated as much as possible without clipping. The beginning and ending image are stuck to the element and whitespace is distributed evenly
Round	The image is repeated and squished or stretched to fill the space (no gaps)
Initial	Sets the property to its default value.
Inherit	Takes the property from its parent element.

Border properties allow developers to change a border: color **(border-color),** width **(border-width)** and style of an specified border (solid, dashed, dotted, etc.).

When styling in CSS, constructing styles that are intended for responsive design or layout, creates the needed user experience and user interface. Let's take a dive into some of the properties and attributes that help developers create responsive layouts for mobile devices, such as: cellphones, tablets, laptops and more.

As you work on responsive layout, 3 key areas take your focus. These are **units of measurement**, **breakpoints and grids** and the **viewport and media query**.

Units of measurement are needed to create the proper relevant size of elements and sections. These include **percentage**, which is relative to the parent element. **Pixel**, which is an absolute element. **Emphasized text (em**), which is relative to the font size of the element itself. The viewports are connected through **view-width (vw)** relative to the viewport width and **view-height (vh)** relative to the viewport height.

Using ***frameworks and templates*** like Bootstrap or Foundation help offer responsive design with prebuilt design capabilities.

Viewports and media queries help with styling the visual areas of your webpage. ***Breakpoints and grids*** are needed to organize layouts. Breakpoints are specific points where the layout of the webpage will respond to provide the best user experience. ***Grids*** help organize content into rows and columns, to ease responsive design.

Utilizing these properties to create responsive layouts to enhance your presentation will assist in smooth understanding of your content.

Accessibility, Readability and Testing

Creating easy to test, read and use code is a industry practice that should be worked on in the beginning stages of learning how to code or when increasing your developer skills. This includes commenting, separation of HTML and external documents, color contrasts and usage, errors, and cascading issues.

When focusing on accessibility, we must construct well-formed HTML and CSS markup to reflect the best industry practices. Let's take a dive into some of the best practices with rules, commenting, web safe fonts, cross-platform usability, and proper separation of HTML & CSS doucments.

When working with rule sets, you can ***reuse rules*** and ***rule sets***. You want to utilize CSS rules and selectors with efficiency to prevent redundancy and increase maintenance ability in your doucments. For example, if you are developing multiple web pages and want to keep a consistent style throughout your website. You have elements such as headings, links, paragraphs etc. that need to look the same. Instead of writing CSS rules for each element on every page or creating a new CSS document for each page, you can simply use reusable classes to define common styles and apply them wherever they are needed.

Creating web applications or software is unique with each developer. This means that each developer's coding / programming styles are different. Adding comments to your HTML and CSS code is needed to explain the purpose of certain code blocks or sections. This is needed to help other developers understand your code and make maintaining it easier. You may include a difficult to understand script in HTML or keyframe in your CSS code that another developer would have a hard time understanding. Inputting a comment with a detailed explanation of the code would help them to see what was being done.

To create a comment in HTML you will type in your document **<! - - comment text - - >**

```
<!-- This is a comment in HTML. -->
```

To create a comment in CSS you will type in your document **/* comment text */**

```
/* This is a comment in CSS */
```

To create a comment in JavaScript you will type in your document **// comment text**

```
// This is a comment in JavaScript
```

Developing versatile websites are more than just beautiful styling and well-structured HTML doucments. Including **web safe fonts** that are available across all operating systems and browsers will help with consistent rendering in your website. Some examples of web safe fonts are Arial, Times New Roman, Helvetica and Georgia. There are many more, but these are some of the most used.

Creating websites is beyond the desktop. Being able to create functionality across various devices and platforms should be included in your usability testing . Modern web browsers allow you to test multiple responsive web design sizes in the inspect element. Optimizing your site for different screen sizes should always be considered when developing.

Our accessibility is also influenced by the HTML and CSS separation. Following the basic principles and concerns of separating your HTML and CSS code will help maintain and scale your project. Using external CSS files to define styles is a great way to promote this.

When focusing on accessibility, taking into consideration accessibility concepts should be included. Let's take a dive into some of the best practices alternative text, color contrast and usage, legibility of typography, tab order, text resizing, text hierarchy and translation.

Providing **alternative text** for images using the alt attribute, creates accessibility for users relying on screen readers. The alt attribute can be used to describe the functions of buttons and form inputs as well. When dealing with **color contrast and usage**, take into consideration users who may have visual impairments. Creating sufficient contrast between element backgrounds and text will help enhance readability. Try to avoid relying on color that expresses important information or functionality. **Legibility of text** is vital. Choose easily readable font family styles, sizes, and line heights. Try avoiding fonts that are difficult to read, small or low in contrast.

Try maintaining a logical tab order to increase the navigation of the website with the keyboard alone. Pay attention to the natural tab order in HTML markup.

Allowing users to resize text without disrupting the layout or functionality of the website is an option. Avoid fixed font sizes and using relative units like em or percentage. Utilize semantic HTML elements such as headings to convey the hierarchy and structure of content, aiding navigation, and comprehension for all users.

In addition, providing options for translating content into different languages to accommodate users from diverse linguistic backgrounds, and consider using language attributes (lang) in HTML to indicate the language of the content.

When focusing on accessibility, following proper structure and integrity of HTML and CSS documents ensures you compile with industry standards. Let’s take a dive into some of the best syntax errors, tag mismatching and cascading issues.

To ensure the quality and compatibility of your web development project, it's good practice to utilize HTML and CSS validators, which inspect your code for syntax errors and adherence to standards. Quick resolutions of your document’s syntax errors are essential to prevent glitches and uphold the integrity of your code. For example, tag mismatches must be corrected diligently to ensure proper nesting and closure of elements, preventing rendering inconsistency and enhance accessibility. Furthermore, gaining proficiency in CSS specificity and inheritance principles helps alleviate unintentional styling conflicts, while the utilization of CSS reset or normalize stylesheets establishes a consistent baseline appearance across various browsers, ultimately improving cross-browser compatibility and enhancing the overall user experience.

Some common HTML and CSS validators available are:

- CSS Validation Service
- CSS Lint
- W3C Markup Validation Service
- Validator.nu – HTML5 Validation Tool

Outro

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 in this book that will help aid your career in software engineering.

Headings & Comments

HTML includes six levels of headings ranked from 1 to 6. Ensure you properly use tags by opening and closing them. For practice, create an H1 and H3 element inside the body tag, displaying your name and new career title. Integrate comments to enhance code understanding.

Elements

All HTML documents consist of HTML elements, starting with an opening tag and ending with a closing tag. Learn about block and inline elements, paragraphs, attributes, images, lists, tables, links, and more. Enhance your coding skills and build a foundation for web development.

Forms

Collecting user information is crucial. Learn about form elements, input types, and attributes. Explore methods like GET and POST, and discover the importance of the action attribute. Incorporate forms into your webpage, ensuring a user-friendly and interactive experience.

Colors, Audio & Video

Explore HTML colors using hexadecimal characters. Add vibrancy to your webpage by changing the background color. Delve into embedding audio and video elements, understanding attributes like controls, loop, and autoplay. Enhance user engagement with multimedia elements.

Unveiling CSS

Welcome to the world of Cascading Style Sheets (CSS)! As you delve into the intricacies of styling, remember the power of rules and selectors. Whether you're defining styles for headers, mastering class and ID selectors, or exploring text styling, your journey is filled with creativity.

The CSS journey covers everything from the box model to background styling, list styling, link styling, table styling, form styling, layout styling, positioning styling, transforms, transitions, and animations. Each concept adds a layer to your coding expertise, transforming you into a versatile developer.

Remember, the flex layout and positioning properties give you control, and transformations open a realm of possibilities. Dive into transitions and animations to bring life to your creations.

Thank You for Choosing Astro Clare Technology!

At Astro Clare Technology, we express our gratitude for choosing to embark on this coding journey with us! Whether you're a dedicated learner or a parent nurturing the next software engineer, your trust means the world to us.

Our mission extends beyond teaching coding skills. We aim to cultivate critical thinking, logical outlooks, problem-solving skills, discipline, and career-ready structures. The evolving landscape of technology should be accessible to all, and we're here to make that happen.

As you progress through the activities, each successful assessment earns you a certificate from Astro Clare Technology, a testament to your accomplishments. Your journey is not just about learning to code; it's about becoming the greatest engineer you can be.

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**.**

Your Journey, Your Achievement

Completing this book marks a significant milestone in your coding journey. Your commitment to learning and growing is commendable. As you showcase your best code forward, know that Astro Clare Technology is proud to be a part of your success.

Clarence Scott,

CEO & Founder

Astro Clare Technology