IDX G9 Computer Essential (S) Study Guide Issue 5: A Comprehensive Guide to HTML and CSS

Gordon. H ©Indexademics

1 Introduction to HTML and CSS

This study guide, brought to you by Indexademics, provides a comprehensive introduction to HTML (HyperText Markup Language) and CSS (Cascading Style Sheets), the fundamental technologies for building web pages. HTML is used to structure the content of a webpage, while CSS is used to style and visually format that content. This guide will cover essential aspects of both HTML and CSS, from basic syntax to more advanced concepts, enabling you to create well-structured and visually appealing web content. We will also delve into some less commonly used but equally important HTML elements to broaden your understanding and capabilities.

2 HTML: Structuring Web Content

HTML is the foundation of all web pages. It uses tags to structure content and define its meaning within a document.

2.1 Basic HTML Document Structure

Every HTML document begins with a basic structure. Understanding this structure is crucial for writing valid HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Page Title</title>
    <link rel="stylesheet" href="styles.css"> <!-- Linking to external CSS file -->
</head>
<body>
    <!-- Visible page content starts here -->
    <h1>Welcome to My Webpage</h1>
    <p>This is a paragraph of text.</p>
</body>
</html>
```

Listing 1: Basic HTML Document Structure

- `<!DOCTYPE html>`: Declares the document type and version of HTML. Always the first line.
- `<html>`: The root element of an HTML page. `lang="en"` specifies the language as English.

- `<head>`: Contains meta-information about the HTML document, not displayed on the page itself.
  - `<meta charset="UTF-8">`: Sets character encoding to UTF-8, supporting most characters.
  - `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Configures viewport for responsive design.
  - `<title>`: Sets the title that appears in the browser tab.
  - `<link rel="stylesheet" href="styles.css">`: Links to an external CSS stylesheet named "styles.css".
- `<body>`: Contains all the visible content of the HTML document.

2.2 Essential HTML Elements

HTML elements are the building blocks of web pages. They are defined by tags.

2.2.1 Headings: `<h1>` to `<h6>`

Headings define the hierarchy of content, with `<h1>` being the most important and `<h6>` the least.

```
<h1>Main Title of the Page</h1>
<h2>Major Section Heading</h2>
<h3>Sub-section Heading</h3>
<h4>Further Sub-section</h4>
<h5>Minor Heading</h5>
<h6>Least Important Heading</h6>
```

Listing 2: HTML Headings

2.2.2 Paragraphs: `<p>`

Paragraphs are used for blocks of text. Browsers automatically add space before and after paragraphs.

```
<p>This is the first paragraph. It contains sentences that form a coherent block of text. Paragraphs
    are essential for presenting textual content in a readable format.</p>
<p>This is another paragraph. Each paragraph should represent a distinct block of information or
    thought.</p>
```

Listing 3: HTML Paragraphs

2.2.3 Line Breaks: `<br>`

The `<br>` tag inserts a single line break. It is useful for writing addresses or poems where line breaks are significant. It's an empty element, meaning it has no closing tag.

```
<p>This is a line of text.<br>
This line will start on a new line because of the <br> tag.</p>

<address>
Indexademics<br>
123 Main Street<br>
Anytown, USA
</address>
```

Listing 4: HTML Line Breaks

2.2.4 Horizontal Rules: `<hr>`

The `<hr>` tag represents a thematic break in content and is often displayed as a horizontal rule. It can be used to separate sections of content visually. Like `<br>`, it is also an empty element.

```
<h1>Section 1</h1>
<p>Content for section 1...</p>

<hr>

<h1>Section 2</h1>
<p>Content for section 2...</p>
```

Listing 5: HTML Horizontal Rule

2.2.5 Divisions: `<div>`

As introduced before, `<div>` elements are block-level containers used to group other HTML elements. They are essential for structuring layouts and applying styles to sections of a webpage.

```
<div id="header">
  <p>Website Header Content</p>
</div>

<div id="content">
  <p>Main body content goes here.</p>
</div>

<div id="footer">
  <p>Footer information.</p>
</div>
```

Listing 6: HTML Divisions

2.2.6 Spans: `<span>`

`<span>` elements are inline containers used to group and style inline elements or parts of text within a block of text. They are useful for applying styles to specific words or phrases without creating a line break.

```
<p>This is a sentence with <span style="color:blue;">some words</span> styled differently using a span
    .</p>
```

Listing 7: HTML Spans

2.2.7 Preformatted Text: `<pre>`

The `<pre>` tag defines preformatted text. Text within a `<pre>` element is displayed in a fixed-width font, and whitespace and line breaks are preserved exactly as written in the HTML file. This is useful for displaying code snippets or text that needs to maintain a specific format.

```
<pre>
function myFunction() {
  var message = "Hello, World!";
  alert(message);
}
</pre>
```

Listing 8: HTML Preformatted Text

2.2.8 Code Snippets: `<code>`

The `<code>` tag is used to display snippets of code inline. It is typically used for short code examples within paragraphs. For longer code blocks, consider using `<pre>` in conjunction with `<code>` or using specialized listing environments like `lstlisting` as used in this document.

```
<p>To define a variable in JavaScript, you use the <code>var</code> keyword.</p>

<pre><code language="javascript">
function exampleFunction() {
  // function body here
}
</code></pre>
```

Listing 9: HTML Code Snippets

2.2.9 Quotations: `<blockquote>` and `<q>`

HTML provides tags for representing quotations:

- `<blockquote>`: Used for longer, block-level quotations. Browsers typically render `<blockquote>` elements with indentation.
- `<q>`: Used for short, inline quotations. Browsers typically enclose `<q>` elements in quotation marks.
- `<cite>`: Defines the title of a work (e.g., a book, a song, a movie, a website) and is often used to cite sources.

```html
<blockquote cite="https://www.example.org/quotation-source">
  <p>This is a longer quotation that is set off from the main text. It might span multiple lines.</p>
  <cite>Source of the quotation</cite>
</blockquote>

<p>As <q cite="https://www.example.org/inline-quote">they say, brevity is the soul of wit</q>.</p>

<p><cite>The Hitchhiker's Guide to the Galaxy</cite> by Douglas Adams is a humorous science fiction
    series.</p>
```

Listing 10: HTML Blockquote and Inline Quote

2.2.10 Abbreviations: `<abbr>`

The `<abbr>` tag is used to define an abbreviation or acronym. Providing the full form of the abbreviation in the `title` attribute can be helpful for accessibility and SEO.

```html
<p>The <abbr title="World Wide Web Consortium">W3C</abbr> develops web standards.</p>
<p>HTML, or <abbr title="HyperText Markup Language">HTML</abbr>, is the standard markup language for
    documents designed to be displayed in a web browser.</p>
```

Listing 11: HTML Abbreviations

2.2.11 Addresses: `<address>`

The `<address>` tag defines contact information for the author or owner of a document or an article. It can include physical addresses, email addresses, phone numbers, and social media links. Browsers typically render `<address>` elements in italics.

```html
<address>
Written by John Doe.<br>
Visit us at:<br>
Indexademics<br>
123 Main Street, Anytown, USA<br>
<a href="mailto:info@indexademics.com">Email Us</a><br>
<a href="tel:+1234567890">Call Us</a>
</address>
```

Listing 12: HTML Address

2.2.12 Emphasis and Strong Importance: `<em>` and `<strong>`

These tags are used for semantic emphasis:

- `<em>` (Emphasis): Represents emphasized text. Browsers typically display emphasized text in italics. It indicates stress emphasis.
- `<strong>` (Strong Importance): Represents strongly important text. Browsers typically display strong text in bold. It indicates strong importance, seriousness, or urgency.

```html
<p>This is <em>emphasized</em> text.</p>
<p>This is <strong>strongly important</strong> text.</p>
<p><strong>Warning:</strong> This action cannot be undone.</p>
<p>Remember to <em>always</em> backup your data.</p>
```

Listing 13: HTML Emphasis and Strong

2.2.13 Small Text: `<small>`

The `<small>` tag defines smaller text. It is often used for side comments, disclaimers, legal restrictions, or copyright information.

```html
<p>This is regular text. <small>This is small text, often used for fine print.</small></p>

<footer>
  <p>  2025 Indexademics <small>All rights reserved.</small></p>
</footer>
```

Listing 14: HTML Small Text

2.2.14 Deleted and Inserted Text: <del> and <ins>

These tags are used to indicate edits to a document:

- <del> (Deleted Text): Represents text that has been deleted from a document. Browsers typically render deleted text with a strikethrough.
- <ins> (Inserted Text): Represents text that has been inserted into a document. Browsers typically render inserted text as underlined.

```
<p>My favorite color is <del>blue</del> <ins>green</ins>.</p>

<p>We are offering a discount of <del>$20</del> <ins>$15</ins> for a limited time!</p>
```

Listing 15: HTML Deleted and Inserted Text

2.2.15 Superscript and Subscript Text: <sup> and <sub>

These tags are used for typographical conventions:

- <sup> (Superscript): Renders text as superscript, raised above the normal line. Used for exponents, ordinal numbers, etc.
- <sub> (Subscript): Renders text as subscript, lowered below the normal line. Used for chemical formulas, mathematical subscripts, etc.

```
<p>E = mc<sup>2</sup></p>
<p>H<sub>2</sub>O is the chemical formula for water.</p>
<p>1<sup>st</sup> Place</p>
```

Listing 16: HTML Superscript and Subscript

2.2.16 Keyboard Input, Sample Output, and Variables: <kbd>, <samp>, <var>

These tags are used to denote different types of computer-related text:

- <kbd> (Keyboard Input): Represents user input from a keyboard. Browsers typically render it in a monospace font.
- <samp> (Sample Output): Represents sample output from a computer program or script. Also typically rendered in a monospace font.
- <var> (Variable): Represents a variable in a mathematical expression or programming context. Often rendered in italics.

```
<p>Press <kbd>Ctrl</kbd> + <kbd>S</kbd> to save the document.</p>
<p>The program output was: <samp>Error: File not found.</samp></p>
<p>In the equation <var>y</var> = <var>mx</var> + <var>b</var>, <var>x</var> and <var>y</var> are
    variables.</p>
```

Listing 17: HTML Keyboard Input

2.2.17 Definition Term: <dfn>

The <dfn> tag is used to define a definition term. It typically appears within a context where a term is being introduced or defined.

```
<p><dfn>HTML</dfn> stands for Hyper Text Markup Language. It is the standard markup language for
    creating web pages.</p>

<p>In computer science, <dfn>algorithm</dfn> refers to a step-by-step procedure for solving a problem.
    </p>
```

Listing 18: HTML Definition Term

2.2.18 Marked Text: <mark>

The <mark> tag represents text that is marked or highlighted for reference purposes, due to its relevance in a particular context. Browsers typically render marked text with a yellow background by default.

```
<p>In this document, <mark>HTML elements</mark> are discussed in detail.</p>
<p>The search results for "<mark>web development</mark>" are shown below.</p>
```

Listing 19: HTML Marked Text

2.2.19 Bidirectional Isolate and Override: <bdi> and <bdo>

These tags are related to text directionality:

- <bdi> (Bidirectional Isolate): Isolates a part of text that might be formatted in a different direction from the surrounding text. Useful when embedding text with unknown directionality, like user-generated content in multiple languages.
- <bdo> (Bidirectional Override): Overrides the current text direction. Use the dir attribute to specify direction (rtl for right-to-left, ltr for left-to-right).

```
<p>User <bdi>Username with RTL text here</bdi> posted a comment.</p>
<p><bdo dir="rtl">This text will be displayed from right to left.</bdo></p>
```

Listing 20: HTML BDI and BDO

2.2.20 Word Break Opportunity: <wbr>

The <wbr> tag specifies where a word could be broken if the line needs to wrap. This is useful for long words that might overflow their container.

```
<p>This is a verylongwordthatmightoverflowitscontainer<wbr>ifitdoesnothaveabreak.</p>
<p>Antidisestablishmentarianism<wbr>is a very long word.</p>
```

Listing 21: HTML Word Break Opportunity

2.2.21 Details and Summary: <details> and <summary>

The <details> tag creates a collapsible interactive widget. The <summary> tag defines a heading for the <details> element, which is visible when the details are collapsed.

```
<details>
  <summary>Click to see details</summary>
  <p>Here are some details that are hidden by default and can be revealed by the user.</p>
</details>

<details>
  <summary>Copyright Information</summary>
  <p>   2023 Indexademics</p>
</details>
```

Listing 22: HTML Details and Summary

2.2.22 Dialog Box or Window: <dialog>

The <dialog> tag represents a dialog box or other interactive component, such as an inspector or window. It can be used to create modal dialogs, alerts, or other pop-up elements. JavaScript is typically used to control the display and behavior of <dialog> elements.

```
<button onclick="document.getElementById('myDialog').showModal();">Open Dialog</button>

<dialog id="myDialog">
  <p>This is a dialog box.</p>
  <button onclick="document.getElementById('myDialog').close();">Close</button>
</dialog>

<script>
// JavaScript to handle dialog functionality can be added here.
</script>
```

Listing 23: HTML Dialog Example

2.2.23 Hyperlinks: <a> (Anchor Tag)

Hyperlinks, or links, connect web pages and resources. The <a> tag creates hyperlinks.

```
<a href="https://indexademics.com/">Visit Indexademics Website</a>
<a href="another_page.html">Go to Another Page on this Site</a>
<a href="#section2">Jump to Section 2 on This Page</a>
<a href="mailto:info@indexademics.com">Email Indexademics</a>
<a href="tel:+1234567890">Call Us</a>
```

Listing 24: HTML Hyperlinks

- href Attribute: Specifies the link's destination.

  - Absolute URL: Full web address to an external site (e.g., `https://indexademics.com/`).
  - Relative URL: Path to a page within the same website (e.g., `another_page.html`).
  - Anchor Link: Links to a specific section within the same page, using `#` and the element's `id` (e.g., `#section2`).
  - Email Link: Opens the user's email client to send an email (e.g., `mailto:info@indexademics.com`).
  - Telephone Link: Initiates a phone call (e.g., `tel:+1234567890`).
- `target` Attribute: Defines where to open the link.
  - `_blank`: Opens in a new tab or window.
  - `_self`: Opens in the same tab (default).
  - Other options like `_parent`, `_top` are less commonly used in basic web development.
- `rel` Attribute: Specifies the relationship between the current document and the linked resource. Common values include:
  - `noopener`: For security when opening links in new tabs. Prevents the new page from accessing the opener page via JavaScript. Often used with `target="_blank"`.
  - `nofollow`: Tells search engine crawlers not to follow the link. Used for links to untrusted content or in user-generated content areas.
  - `alternate`: Indicates an alternate version of the document (e.g., a different language or media type).
  - `author`: Link to the author of the current document.
  - `help`: Link to help documentation.
  - `license`: Link to copyright information for the document.
  - `prev` and `next`: Indicate previous and next documents in a series.

2.2.24 Images: `<img>`

The `<img>` tag embeds images into a web page.

```
<img src="images/logo.png" alt="Indexademics Logo" width="200" height="100">
```

Listing 25: HTML Images

- `src` Attribute: Path to the image file.
- `alt` Attribute: Alternative text for the image, for accessibility and if the image fails to load. Crucial for SEO.
- `width` and `height` Attributes: Specify image dimensions in pixels. Best practice is to control sizing with CSS.
- `loading` Attribute: Controls how the browser loads the image, improving page load performance.
  - `lazy`: Defers loading the image until it is about to enter the viewport. Good for performance, especially for long pages with many images.
  - `eager`: Loads the image immediately, regardless of whether it is in the viewport. Default behavior.
  - `auto`: Browser decides whether to load eagerly or lazily.
- `srcset` and `sizes` Attributes: For responsive images, allowing the browser to choose the most appropriate image source based on screen size and resolution. More advanced topic for responsive image handling.

2.2.25 Lists: `<ul>`, `<ol>`, `<li>`

HTML supports unordered and ordered lists.

- Unordered Lists (`<ul>`): For lists where order doesn't matter, using bullet points.
- Ordered Lists (`<ol>`): For lists where order is important, using numbers or letters.
- List Items (`<li>`): Used within `<ul>` or `<ol>` to define each item.

```html
<!-- Unordered List -->
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>

<!-- Ordered List -->
<ol>
  <li>First Step</li>
  <li>Second Step</li>
  <li>Third Step</li>
</ol>

<!-- Nested List -->
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Listing 26: HTML Lists

2.2.26 Tables: `<table>`, `<tr>`, `<th>`, `<td>`

Tables display data in rows and columns.

```html
<table>
  <caption>Example Data Table</caption>
  <thead>
    <tr>
      <th>Column 1 Header</th>
      <th>Column 2 Header</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Row 1, Data 1</td>
      <td>Row 1, Data 2</td>
    </tr>
    <tr>
      <td>Row 2, Data 1</td>
      <td>Row 2, Data 2</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="2">Table Footer Information</td>
    </tr>
  </tfoot>
</table>
```

Listing 27: HTML Tables

- `<table>`: Defines the table container.
- `<caption>`: Optional caption describing the table's content.
- `<thead>`: Table header section, typically containing column headers.
- `<tbody>`: Table body section, containing the main table data.
- `<tfoot>`: Table footer section, often for summaries or notes.
- `<tr>` (Table Row): Defines a row within a table.
- `<th>` (Table Header): Defines a header cell, usually bold and centered.
- `<td>` (Table Data): Defines a standard data cell.
- `colspan` and `rowspan` attributes in `<td>` or `<th>` can make cells span multiple columns or rows.
- `scope` attribute in `<th>`: Defines whether the header cell is a header for a column, row, or group of rows or columns. Improves accessibility, especially for screen readers. Common values: `col`, `row`, `colgroup`, `rowgroup`.
- `<colgroup>` and `<col>`: Used to define column groups and column properties for styling or semantic purposes.

2.2.27 Forms: `<form>`, `<input>`, `<textarea>`, `<button>`, `<select>`, `<label>`

Forms are used to collect user input.

```
<form action="/submit-form" method="post">
  <label for="name">Name:</label><br>
  <input type="text" id="name" name="user_name" required><br><br>

  <label for="email">Email:</label><br>
  <input type="email" id="email" name="user_email"><br><br>

  <label for="message">Message:</label><br>
  <textarea id="message" name="user_message" rows="4" cols="50"></textarea><br><br>

  <input type="radio" id="option1" name="options" value="1">
  <label for="option1">Option 1</label><br>
  <input type="radio" id="option2" name="options" value="2">
  <label for="option2">Option 2</label><br><br>

  <select id="dropdown" name="dropdown_option">
    <option value="value1">Option 1</option>
    <option value="value2">Option 2</option>
  </select><br><br>

  <input type="checkbox" id="agree" name="agreement" value="agree">
  <label for="agree">I agree to terms</label><br><br>

  <button type="submit">Submit</button>
</form>
```

Listing 28: HTML Forms

- `<form>`: Defines an HTML form.
  - `action` Attribute: URL to process form data.
  - `method` Attribute: HTTP method to submit form data (`get` or `post`, `post` is generally preferred for forms with sensitive data or large amounts of data).
  - `enctype` Attribute: Specifies how form data should be encoded when submitting to the server. Important for file uploads (`enctype="multipart/form-data"`).
  - `autocomplete` Attribute: Enables or disables autocomplete for the form. Can be set to `on` or `off`, or specific input fields can override the form setting.
- Input Elements:
  - `<input type="text">`: Single-line text input.
  - `<input type="email">`: Email input, with validation for email format.
  - `<input type="password">`: Password input, text is masked.
  - `<input type="radio">`: Radio buttons, for selecting one option from many.
  - `<input type="checkbox">`: Checkboxes, for selecting one or more options.
  - `<input type="submit">`: Submit button to send form data.
  - `<input type="reset">`: Reset button to clear form inputs.
  - `<input type="file">`: File upload control.
  - `<input type="date">`, `<input type="time">`, `<input type="datetime-local">`, `<input type="number">`, `<input type="range">`, `<input type="color">`: HTML5 input types for specific data types, often with built-in validation and UI elements.
  - `placeholder` Attribute: Provides a hint to the user of the expected input value.
  - `value` Attribute: Sets the initial value of the input field.
  - `readonly` Attribute: Makes the input field read-only.
  - `disabled` Attribute: Disables the input field, making it non-interactive and not submitted with the form.
  - `pattern` Attribute: Specifies a regular expression that the input value must match.
  - `min`, `max`, `step` Attributes: Constraints for number and range inputs.
  - `multiple` Attribute: Allows multiple file selection in file inputs or multiple selections in select elements.
- `<textarea>`: Multi-line text input area.
  - `rows` and `cols` Attributes: Specify the visible height and width of the textarea in characters (can be overridden by CSS).

- `<select>` and `<option>`: Dropdown list.
  - `<optgroup>`: Used to group related options within a select dropdown.
  - `selected` Attribute in `<option>`: Specifies that an option should be pre-selected when the page loads.
- `<label>`: Associates text with form controls, improving accessibility. The `for` attribute should match the `id` of the associated input.
- `<button>`: Represents a clickable button. Can be used to submit forms or trigger other actions with JavaScript.
  - `type` Attribute: Specifies the type of button (`submit`, `reset`, `button`). Defaults to `submit` if inside a form, otherwise `button`.
- `<fieldset>` and `<legend>`: `<fieldset>` groups related form elements, and `<legend>` provides a caption for the `<fieldset>` group. Useful for logical grouping and accessibility.

2.2.28 Semantic HTML Elements

Semantic elements give meaning to the structure, improving accessibility and SEO.

- `<header>`: Introductory content, usually at the top of a page or section.
- `<nav>`: Navigation links.
- `<main>`: Main content of the document.
- `<article>`: Self-contained content, like a blog post or news article.
- `<section>`: Thematic grouping of content within a page.
- `<aside>`: Content related to the main content, but not essential to it (e.g., sidebars).
- `<footer>`: Footer of a document or section, often containing copyright info, contact details.
- `<address>`: Contact information for the author/owner (also listed above but semantically relevant).
- `<figure>` and `<figcaption>`: `<figure>` encapsulates self-contained content like images, illustrations, diagrams, code listings, etc., and `<figcaption>` provides a caption for the `<figure>`.

```
<header>
  <h1>Website Name</h1>
  <nav>
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/about">About</a></li>
      <li><a href="/services">Services</a></li>
      <li><a href="/contact">Contact</a></li>
    </ul>
  </nav>
</header>

<main>
  <article>
    <section>
      <h2>Article Title</h2>
      <p>This is the main content of the article...</p>

      <figure>
        <img src="images/diagram.png" alt="Diagram of something">
        <figcaption>Figure 1: Explanation of the diagram.</figcaption>
      </figure>

    </section>
  </article>

  <aside>
    <h3>Related Articles</h3>
    <ul>
      <li><a href="#">Article 1</a></li>
      <li><a href="#">Article 2</a></li>
    </ul>
  </aside>
</main>

<footer>
  <address>
    Contact us at: <a href="mailto:info@indexademics.com">info@indexademics.com</a>
```

```
  </address>
  <p>   2023 Indexademics</p>
</footer>
```

Listing 29: Semantic HTML Example

2.3 HTML Attributes

Attributes provide additional information about HTML elements. They are included in the start tag.

- Common Attributes:
  - `class`: Specifies one or more class names for CSS styling and JavaScript manipulation.
  - `id`: Unique identifier for an element, used by CSS and JavaScript. Must be unique within a document.
  - `style`: Inline CSS styles applied directly to the element.
  - `title`: Tooltip text that appears when the mouse hovers over the element.
- Global Attributes (can be used on almost any HTML element):
  - `accesskey`: Specifies a keyboard shortcut to activate or focus an element.
  - `draggable`: Specifies whether an element is draggable.
  - `hidden`: Hides an element.
  - `lang`: Language of the element's content (e.g., `lang="es"` for Spanish).
  - `dir`: Text direction (`ltr` for left-to-right, `rtl` for right-to-left).
  - `spellcheck`: Specifies whether the element should be spellchecked.
  - `tabindex`: Specifies the tab order of an element.
  - `contenteditable`: Specifies whether the content of an element is editable by the user.
  - `data-*`: Used to store custom data private to the page or application.
- Event Attributes (JavaScript event handlers, e.g., `onclick`, `onload`, `onsubmit`): Allow JavaScript code to be executed in response to events. Beyond the scope of basic HTML but important for interactive web pages.

2.4 HTML Media: Audio and Video

HTML provides elements to embed multimedia content like audio and video directly into web pages.

2.4.1 HTML Audio: `<audio>`

The `<audio>` element is used to embed sound content in an HTML document.

```
<audio controls>
  <source src="audio/sample.mp3" type="audio/mpeg">
  <source src="audio/sample.ogg" type="audio/ogg">
  Your browser does not support the audio element.
</audio>
```

Listing 30: HTML Audio Example

- `<audio>`: Defines the audio container.
  - `controls` Attribute: Displays audio controls (play, pause, volume, etc.). Essential for user interaction.
  - `autoplay` Attribute: Starts playing the audio automatically (use with caution as it can be intrusive).
  - `loop` Attribute: Loops the audio playback.
  - `muted` Attribute: Starts the audio muted.
  - `preload` Attribute: Specifies if and how the audio should be loaded when the page loads (`auto`, `metadata`, `none`).
- `<source>`: Allows specifying multiple audio files for different browser support.
  - `src` Attribute: Path to the audio file.
  - `type` Attribute: MIME type of the audio file (e.g., `audio/mpeg`, `audio/ogg`, `audio/wav`).
- Text content within `<audio>` tag: Displayed if the browser does not support the `<audio>` element.
- `crossorigin` Attribute: Handles CORS (Cross-Origin Resource Sharing) for audio resources from different domains.
- `volume` Property (JavaScript): Allows controlling volume programmatically.
- `currentTime` Property (JavaScript): Allows getting or setting the current playback position.

2.4.2 HTML Video: `<video>`

The `<video>` element is used to embed video content in an HTML document.

```
<video width="640" height="360" controls poster="images/video-poster.jpg">
  <source src="videos/sample.mp4" type="video/mp4">
  <source src="videos/sample.webm" type="video/webm">
  <track src="subtitles_en.vtt" kind="subtitles" srclang="en" label="English">
  Your browser does not support the video element.
</video>
```

Listing 31: HTML Video Example

- `<video>`: Defines the video container.
  - `controls` Attribute: Displays video controls (play, pause, volume, fullscreen, etc.).
  - `width` and `height` Attributes: Set the dimensions of the video player in pixels.
  - `poster` Attribute: Specifies an image to be shown while the video is downloading or until the user starts playing.
  - `autoplay` Attribute: Starts playing the video automatically (use with caution).
  - `loop` Attribute: Loops the video playback.
  - `muted` Attribute: Starts the video muted.
  - `preload` Attribute: Specifies if and how the video should be loaded when the page loads (`auto`, `metadata`, `none`).
  - `playsinline` Attribute: For mobile devices, prevents video from automatically going fullscreen on playback start.
  - `crossorigin` Attribute: Handles CORS for video resources from different domains.
  - `disablePictureInPicture` Attribute: Disables the Picture-in-Picture mode for the video.
  - `disableRemotePlayback` Attribute: Prevents remote playback (e.g., casting to a TV).
- `<source>`: Allows specifying multiple video files for different browser support.
  - `src` Attribute: Path to the video file.
  - `type` Attribute: MIME type of the video file (e.g., `video/mp4`, `video/webm`, `video/ogg`).
- `<track>`: Used to add subtitles, captions, or other text tracks to the video.
  - `src` Attribute: Path to the track file (e.g., WebVTT - `.vtt` files).
  - `kind` Attribute: Specifies the kind of track (`subtitles`, `captions`, `descriptions`, `chapters`, `metadata`).
  - `srclang` Attribute: Language of the track content (e.g., `en` for English).
  - `label` Attribute: User-readable title for the track.
  - `default` Attribute: Specifies that the track should be enabled by default if no user language preference is indicated.
- Text content within `<video>` tag: Displayed if the browser does not support the `<video>` element.
- JavaScript API: The `<video>` element has a rich JavaScript API for controlling playback, getting metadata, handling events, and more, enabling advanced video player functionality.

3 CSS: Styling Web Pages

CSS (Cascading Style Sheets) is used to control the presentation of HTML elements, including layout, colors, fonts, and more.

3.1 CSS Syntax

CSS rules consist of a selector and a declaration block.

```
selector {
  property: value;
  property: value;
  /* more declarations */
}
```

Listing 32: CSS Syntax

- Selector: Targets the HTML element(s) to be styled.
- Declaration Block: Enclosed in curly braces {}. Contains one or more declarations.
- Declaration: Consists of a CSS property and a value, separated by a colon `:`, and each declaration ends with a semicolon `;`.

3.2 CSS Selectors

CSS selectors target specific HTML elements to apply styles.

3.2.1 Element Selectors

Selects all HTML elements of a given type.

```
p {
  color: #333; /* Dark gray color */
  font-family: sans-serif;
  line-height: 1.6;
}
```

Listing 33: Element Selector Example

Styles all `<p>` elements to be dark gray, sans-serif font, and with a line height of 1.6.

3.2.2 Class Selectors

Selects elements with a specific `class` attribute. Class selectors start with a dot `.`

```
.highlight-text {
  background-color: yellow;
  padding: 5px;
  font-weight: bold;
}
```

Listing 34: Class Selector Example

Styles elements with `class="highlight-text"` to have a yellow background, padding, and bold font.

3.2.3 ID Selectors

Selects the element with a specific `id` attribute. ID selectors start with a hash `#`. IDs should be unique.

```
#main-navigation {
  background-color: #f9f9f9;
  border-bottom: 1px solid #ddd;
  padding: 10px 0;
}
```

Listing 35: ID Selector Example

Styles the element with `id="main-navigation"` with a light background and bottom border.

3.2.4 Universal Selector

Selects all HTML elements. Represented by an asterisk `*`

```
* {
  box-sizing: border-box; /* Include padding and border in element's total width and height */
  margin: 0;
  padding: 0;
}
```

Listing 36: Universal Selector Example

Resets margin and padding for all elements and sets `box-sizing` to `border-box`.

3.2.5 Attribute Selectors

Selects elements based on the presence or value of an attribute.

```
input[type="text"] {
  border: 1px solid #ccc;
  border-radius: 3px;
  padding: 8px;
}

a[target="_blank"] { /* Selects links with target="_blank" */
  color: blue;
}
```

Listing 37: Attribute Selector Example

Styles text input fields and links that open in a new tab.

3.2.6 Pseudo-class Selectors

Selects elements based on state or position. Pseudo-classes start with a colon `:`.

```
a:hover { /* Style links on mouse hover */
  color: red;
  text-decoration: none;
}

button:focus { /* Style button when focused (e.g., after clicking) */
  outline: 2px solid blue;
}

/* Structural pseudo-classes */
li:first-child { /* First list item */
  font-weight: bold;
}

li:last-child { /* Last list item */
  border-bottom: none;
}

p:nth-child(odd) { /* Every odd paragraph */
  background-color: #f0f0f0;
}

p:nth-child(even) { /* Every even paragraph */
  background-color: #e0e0e0;
}

/* UI state pseudo-classes */
input:enabled { /* Styles enabled input fields */
  border-color: green;
}

input:disabled { /* Styles disabled input fields */
  background-color: #ddd;
  color: #999;
}

input:checked + label { /* Styles label next to a checked checkbox */
  font-weight: bold;
}
```

Listing 38: Pseudo-class Selector Example

Styles links on hover, buttons when focused, and demonstrates structural and UI state pseudo-classes.

3.2.7 Pseudo-element Selectors

Selects specific parts of an element. Pseudo-elements start with a double colon `::`.

```
p::first-line { /* Style the first line of paragraphs */
  font-weight: bold;
}

ul::before { /* Insert content before unordered lists */
  content: "List: ";
  font-weight: bold;
}

/* ::marker - styles the bullet or number of a list item */
li::marker {
  color: blue;
  font-size: 1.2em;
}

/* ::selection - styles the text that is selected by the user */
::selection {
  background-color: #ffcc80; /* Light orange */
  color: white;
}

/* ::placeholder - styles the placeholder text in input fields */
input::placeholder {
  color: #aaa;
  font-style: italic;
}
```

Listing 39: Pseudo-element Selector Example

Styles the first line of paragraphs, adds "List: " before unordered lists, and demonstrates styling list markers, text selection, and input placeholders.

3.2.8 Combinators

Combinators define the relationship between selectors.

- Descendant selector (space): Selects all descendants of an element.
- Child selector (>): Selects only direct children of an element.
- Adjacent sibling selector (+): Selects the immediately following sibling element.
- General sibling selector (~): Selects all sibling elements that follow after.

```
/* Descendant selector */
div p { /* Selects all <p> elements inside <div> elements */
  color: green;
}

/* Child selector */
ul > li { /* Selects only direct <li> children of <ul> */
  list-style-type: circle;
}

/* Adjacent sibling selector */
h2 + p { /* Selects the first <p> immediately after an <h2> */
  margin-top: 0;
}

/* General sibling selector */
div ~ p { /* Selects all <p> siblings of a <div> (that come after the div) */
  font-size: 0.9em;
}
```

Listing 40: Combinator Selector Example

Demonstrates different combinators and their effects on element selection.

3.3 Methods to Include CSS in HTML

There are three main ways to incorporate CSS into your HTML documents.

3.3.1 Inline CSS

Applying styles directly within HTML tags using the `style` attribute.

```
<p style="color: green; font-size: 16px;">This paragraph is styled inline.</p>
```

Listing 41: Inline CSS Example

Pros: Quick for testing or very specific, one-off styles. **Cons:** Not maintainable for larger projects, difficult to reuse styles, mixes content with presentation. Generally discouraged for extensive styling.

3.3.2 Internal CSS (Embedded CSS)

CSS rules are defined within the `<style>` tag inside the `<head>` section of the HTML document.

```
<!DOCTYPE html>
<html>
<head>
<title>Internal CSS Example</title>
<style>
  p {
    color: blue;
    font-size: 16px;
  }
</style>
</head>
<body>
  <p>This paragraph is styled using internal CSS.</p>
</body>
</html>
```

Listing 42: Internal CSS Example

Pros: Useful for page-specific styles. Keeps CSS and HTML in one file, which can be convenient for single-page documents. **Cons:** Less maintainable than external stylesheets for larger sites, still mixes structure and presentation to some extent.

3.3.3 External CSS (Linked CSS)

CSS rules are written in a separate `.css` file and linked to the HTML document using the `<link>` tag in the `<head>` section. This is the recommended method for most projects.

- Create a file, e.g., `styles.css`, and write CSS rules in it.
- Link it in your HTML `<head>`:

HTML file:

```
<!DOCTYPE html>
<html>
<head>
  <title>External CSS Example</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <p>This paragraph is styled using external CSS.</p>
</body>
</html>
```

Listing 43: External CSS Link in HTML

styles.css file:

```
p {
  color: blue;
  font-size: 16px;
}
```

Listing 44: External CSS File Content

Pros: Best for maintainability, reusability, and organization. Clear separation of content (HTML) and presentation (CSS). Easier to update styles across a website. Improves website performance through caching of CSS files. **Cons:** Requires managing separate files.

3.4 Common CSS Properties

CSS properties control various aspects of element styling.

3.4.1 Text Properties

- `color`: Text color (e.g., `color: red;`).
- `font-family`: Font (e.g., `font-family: Arial, sans-serif;`).
- `font-size`: Text size (e.g., `font-size: 16px;`).
- `font-weight`: Boldness (e.g., `font-weight: bold;`).
- `font-style`: Italic or normal (`font-style: italic;`).
- `text-align`: Horizontal alignment (`text-align: center;`).
- `text-decoration`: Underline, overline, etc. (`text-decoration: underline;`).
- `line-height`: Vertical space between lines (e.g., `line-height: 1.5;`).
- `letter-spacing`: Space between characters (e.g., `letter-spacing: 1px;`).
- `word-spacing`: Space between words (e.g., `word-spacing: 5px;`).
- `text-transform`: Uppercase, lowercase, capitalize (`text-transform: uppercase;`).
- `text-indent`: Indentation of the first line of text (`text-indent: 20px;`).
- `white-space`: Handles whitespace and line wrapping (`white-space: nowrap;`).
- `text-shadow`: Adds shadow to text (`text-shadow: 2px 2px 3px 333;`).

3.4.2 Background Properties

- `background-color`: Background color (e.g., `background-color: eee;`).
- `background-image`: Background image (e.g., `background-image: url("image.png");`).
- `background-repeat`: Image repeat behavior (`background-repeat: no-repeat;`).
- `background-position`: Image position (`background-position: center;`).
- `background-size`: Image size (`background-size: cover;`, `contain`, `100px 50px`, etc.).
- `background-origin`: Origin of the background image (`padding-box`, `border-box`, `content-box`).
- `background-clip`: Clipping of the background (`border-box`, `padding-box`, `content-box`, `text`).
- `background-attachment`: Background scrolling behavior (`scroll`, `fixed`, `local`).
- `background`: Shorthand property for setting multiple background properties in one declaration.

3.4.3 Box Model Properties

The CSS box model describes the rectangular boxes that represent HTML elements.

- `margin`: Space outside the element's border (e.g., `margin: 10px;`, `margin-top`, `margin-right`, `margin-bottom`, `margin-left`). Can use `auto` for horizontal centering.
- `padding`: Space inside the element's border, between border and content (e.g., `padding: 20px;`, `padding-top`, `padding-right`, `padding-bottom`, `padding-left`).
- `border`: Element's border (e.g., `border: 1px solid black;`). Properties like `border-width`, `border-style` (`solid`, `dashed`, `dotted`, `double`, etc.), `border-color`. Also `border-radius` for rounded corners, and individual border properties like `border-top`, `border-right`, etc.
- `width`: Element's content width (e.g., `width: 300px;`, `width: 100%`).
- `height`: Element's content height (e.g., `height: 150px;`, `height: auto`).
- `box-sizing`: How width and height are calculated (`box-sizing: border-box;` includes padding and border in width/height, `content-box` - default, width/height only applies to content).
- `overflow`: Handles content overflow (`overflow: hidden;`, `scroll;`, `auto;`, `visible;`). Also `overflow-x` and `overflow-y`.
- `box-shadow`: Adds shadow to boxes (`box-shadow: 5px 5px 10px 888;`).
- `opacity`: Sets the transparency of an element (`opacity: 0.5;`).
- `visibility`: Controls element visibility (`visibility: hidden;` - element is hidden but takes up space, `visibility: visible;`, `visibility: collapse;`).

3.4.4 Display Properties

Controls how elements are displayed and laid out.

- `display: block;`: Element takes up full width, starts a new line (like `<div>`, `<p>`).
- `display: inline;`: Element takes only necessary width, does not start a new line (like `<span>`, `<a>`).
- `display: inline-block;`: Inline flow, but can set width and height (like inline, but with block properties).
- `display: none;`: Element is hidden and removed from page flow.
- `display: flex;`: Enables Flexbox layout for flexible and responsive layouts.
- `display: grid;`: Enables Grid layout for two-dimensional layouts.
- `display: list-item;`: Makes an element behave like a list item.
- `display: table;`, `display: table-cell;`, `display: table-row;`, etc.: CSS table layout properties.

3.4.5 Positioning Properties

Controls the position of elements within the document.

- `position: static;`: Default positioning, follows normal document flow.
- `position: relative;`: Position relative to its normal position. Use `top`, `right`, `bottom`, `left` to offset.
- `position: absolute;`: Removed from document flow, positioned relative to nearest positioned ancestor (or `<html>`).
- `position: fixed;`: Removed from document flow, positioned relative to viewport, stays fixed during scrolling.
- `position: sticky;`: Relative until a threshold is met, then becomes fixed (sticks to the viewport during scroll).
- `top`, `right`, `bottom`, `left`: Offsets for positioned elements. Can use pixel values, percentages, or other CSS units.
- `z-index`: Stack order of elements (higher value is in front). Only works on positioned elements (relative, absolute, fixed, sticky).
- `float`: Positions an element to the left or right and allows text and inline elements to wrap around it (`float: left;`, `float: right;`, `float: none;`). Often used for layout, but Flexbox and Grid are generally preferred for modern layouts.
- `clear`: Specifies which sides of an element are not allowed to float next to other floating elements (`clear: both;`, `clear: left;`, `clear: right;`, `clear: none;`). Used to control flow around floated elements.

3.5 CSS Units

Units in CSS define sizes and lengths.

3.5.1 Absolute Units

Fixed sizes, not relative to anything else.

- `px` (pixels): Basic unit, device-dependent. Generally recommended for screen sizes.
- `pt` (points), `pc` (picas), `in` (inches), `cm` (centimeters), `mm` (millimeters): Physical units, less common for screen display, more relevant for print styles.

3.5.2 Relative Units

Sizes relative to other values.

- `%`: Percentage, relative to parent element or viewport. Very versatile for responsive design.
- `em`: Relative to the font size of the parent element. `1em` = current font size. Useful for maintaining proportional scaling relative to text size.
- `rem` (root em): Relative to the font size of the root element (`<html>`). Good for consistent scaling across the entire website based on a single root font size.
- `vw` (viewport width): 1vw = 1
- `vh` (viewport height): 1vh = 1
- `vmin`, `vmax`: Relative to the smaller or larger dimension of the viewport. Useful for elements that need to scale based on the viewport's smallest or largest dimension.
- `ch` (character width): Width of the "0" (zero) character of the current font.
- `ex` (x-height): Height of the "x" character of the current font.

3.6 Styling Lists and Tables with CSS

CSS provides properties to style HTML lists and tables.

3.6.1 List Styling

- `list-style-type`: Bullet type for `<ul>`, numbering for `<ol>` (`disc`, `circle`, `square`, `decimal`, `decimal-leading-zero`, `lower-roman`, `upper-roman`, `lower-alpha`, `upper-alpha`, `none`, etc.).
- `list-style-position`: Marker position (`inside`, `outside`). `outside` is default, markers are outside the content box. `inside` places markers inside, affecting text flow.
- `list-style-image`: Custom image for list marker (`list-style-image: url('bullet.png');`).
- `list-style`: Shorthand property for setting all list-style properties in one declaration.

```
ul {
  list-style-type: square; /* Square bullets */
}

ol {
  list-style-type: upper-roman; /* Uppercase Roman numerals */
}

.custom-list {
  list-style-image: url('images/checkmark.png'); /* Custom image bullet */
  list-style-position: inside;
  padding-left: 20px; /* Adjust padding for inside position */
}
```

Listing 45: List Styling Example

3.6.2 Table Styling

- `border-collapse`: `collapse` (single border) or `separate` (separated borders). `collapse` is often preferred for cleaner table designs.
- `border-spacing`: Space between cell borders (if `border-collapse: separate;`).
- `caption-side`: Caption position (`top`, `bottom`).
- `empty-cells`: Show or hide borders and background of empty table cells (`show`, `hide`).
- `table-layout`: Algorithm used to lay out table cells, rows, and columns (`auto`, `fixed`). `fixed` can be faster for rendering large tables.
- Styling table parts: Use selectors like `table`, `thead`, `tbody`, `tfoot`, `tr`, `th`, `td` to style different parts of the table.
- Zebra striping: Using pseudo-classes like `tbody tr:nth-child(even)` to style alternating rows for readability.

```
table {
  border-collapse: collapse; /* Single borders */
  width: 100%;
}

th, td {
  border: 1px solid #ddd;
  padding: 8px;
  text-align: left;
}

th {
  background-color: #f0f0f0; /* Light gray header background */
  font-weight: bold;
}

tbody tr:nth-child(even) { /* Zebra striping for even rows */
  background-color: #f9f9f9;
}

caption {
  caption-side: bottom; /* Caption below table */
  font-style: italic;
  color: gray;
  padding-top: 10px;
}
```

Listing 46: Table Styling Example

3.7 Responsive Design with CSS Media Queries

Media queries apply different styles based on device characteristics, like screen width.

```
/* For screens smaller than 768px wide (e.g., mobile devices) */
@media screen and (max-width: 768px) {
  body {
    font-size: 14px; /* Smaller font size for mobile */
  }
  .container {
    width: 100%; /* Full width container on mobile */
  }
}

/* For screens 768px wide and larger (e.g., desktops) */
@media screen and (min-width: 769px) {
  body {
    font-size: 16px; /* Default font size for desktop */
  }
  .container {
    width: 960px; /* Fixed width container on desktop */
    margin: 0 auto; /* Center container on desktop */
  }
}

/* Example using orientation */
@media screen and (orientation: portrait) {
  /* Styles for portrait mode (e.g., mobile phone vertically) */
  .menu {
    flex-direction: column; /* Stack menu items vertically in portrait */
  }
}

@media screen and (orientation: landscape) {
  /* Styles for landscape mode (e.g., mobile phone horizontally, tablet, desktop) */
  .menu {
    flex-direction: row; /* Display menu items horizontally in landscape */
  }
}

/* Example using device pixel ratio for high-resolution screens (Retina) */
@media (-webkit-min-device-pixel-ratio: 2), (min-resolution: 192dpi) {
  .logo {
    content: url("logo-2x.png"); /* Use a higher resolution logo for Retina screens */
    width: 200px; /* Adjust width as needed */
  }
}
```

Listing 47: Media Query Example

Media queries use `@media` followed by media type (`screen`, `print`, `all`) and conditions like `max-width`, `min-width`, `orientation`. Can also use features like `device-width`, `device-height`, `aspect-ratio`, `color`, `resolution`, and more. Media queries can be combined using logical operators like `and`, `or`, `not`, and can target different media types (e.g., `@media print { /* print styles here */ }`).

4 HTML Icons

Icons enhance visual communication on web pages.

4.1 Using Icon Libraries

Icon libraries provide collections of pre-made icons.

4.1.1 Font Awesome

A popular icon library.

1. Include Font Awesome CSS in HTML `<head>`:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min
    .css" integrity="sha512-iecdLmaskl7CVkqkXNQ/ZH/XLlvWZOJyj7Yy7tcenmpD1ypASozpmT/
    E0iPtmFIB46ZmdtAc9eNBvH0H/ZpiBw==" crossorigin="anonymous" referrerpolicy="no-referrer" />
```

Listing 48: Font Awesome CSS Link

2. Use icons with `<i>` tags and Font Awesome classes:

```html
<i class="fas fa-home"></i> Home
<i class="fas fa-search"></i> Search
<i class="fas fa-bars"></i> Menu
<i class="fab fa-twitter"></i> Twitter
<i class="fas fa-cog fa-spin"></i> Settings (spinning icon)
```

Listing 49: Font Awesome Icon Usage

4.1.2 Material Icons (Google Icons)

Another widely used icon set.

1. Include Material Icons CSS in HTML `<head>`:

```html
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

Listing 50: Material Icons CSS Link

2. Use icons with `<span>` tags and Material Icons class names:

```html
<span class="material-icons">home</span> Home
<span class="material-icons">search</span> Search
<span class="material-icons">menu</span> Menu
<span class="material-icons">settings</span> Settings
<span class="material-icons">account_circle</span> Profile
```

Listing 51: Material Icons Usage

4.1.3 SVG Icons

Scalable Vector Graphics (SVG) icons can be directly embedded into HTML or used as image files.

- Inline SVG: Embed SVG code directly into HTML.
- SVG Files: Use SVG files as image sources in `<img>` tags or as background images in CSS.
- Pros: Scalable without loss of quality, can be styled with CSS, smaller file sizes compared to raster images for icons.

```html
<svg width="24" height="24" viewBox="0 0 24 24">
  <path fill="currentColor" d="M10 20v-6h4v6h5v-8h3L12 3 2 12h3v8z"/>
</svg> Home
```

Listing 52: Inline SVG Icon Example

5 Conclusion

This study guide has provided a comprehensive overview of HTML and CSS for what we learned at school covering essential concepts from basic structure to styling, responsive design, embedding media, and even less common HTML elements to broaden your knowledge. Understanding and utilizing these technologies is crucial for web development. Continue to practice, explore, and build projects to solidify your skills and deepen your knowledge in creating web content. For further resources and advanced topics, explore the official documentation and online communities dedicated to web development. IDX not only aims to teach you what we learned at school, we also teach you what you should know and things beyond that. Thats the reason why people choose IDX over PA. After all, good luck with your exams! Visit Us: https://indexademics.com/