Lab-1 : Introduction to HTML & CSS

Md Sabbir Hossain, Md Ismail Bhuiyan

Summer 2025

Introduction

This reference guide provides a clear and comprehensive summary of fundamental HTML and CSS concepts. It is tailored for beginners using Visual Studio Code (VS Code) to develop and preview websites. Practical examples, explanations, and code syntax are included for effective learning.

Contents

1 HTML Tags

1.1 Document Declaration: `<!DOCTYPE html>`

The `<!DOCTYPE html>` declaration defines the document type and HTML version. It is not an HTML tag, but an instruction to the browser to render the document in standards-compliant mode (HTML5 in this case).

Example:

```html
<!DOCTYPE html>
```

This must appear at the very beginning of every HTML document, before the `<html>` tag. Omitting it can cause inconsistent rendering across browsers.

1.2 HTML Structure: `<html>`, `<head>`, `<body>`

These tags define the overall structure of an HTML document.

- `<html>` – Root element that wraps all content.
- `<head>` – Contains metadata, links to CSS, and the page title.
- `<body>` – Contains the visible content displayed in the browser.

Example:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>My Website</title>
  </head>
  <body>
    <h1>Welcome</h1>
    <p>This is my website.</p>
  </body>
</html>
```

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

HTML provides six levels of headings, where `<h1>` is the highest (most important) and `<h6>` is the lowest. Headings help structure content semantically and are important for accessibility.

Example:

```html
<h1>Main Title</h1>
<h2>Section Heading</h2>
<h3>Subsection</h3>
<h4>Detail</h4>
<h5>Minor Detail</h5>
<h6>Note</h6>
```

Use only one `<h1>` per page when possible, and follow a logical order to maintain semantic hierarchy.

1.4 Paragraph and Line Break: <p>, <br>

The <p> tag defines a block-level paragraph, and browsers automatically add space before and after it. The <br> tag inserts a single line break and does not require a closing tag.

Example:

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

<p>Line one.<br>Line two (same paragraph).</p>
```

Avoid excessive use of <br> for layout—use CSS for spacing when possible.

1.5 Text Formatting Tags: <b>, <strong>, <i>, <em>, <u>, <mark>, <small>, <del>, <ins>

HTML provides several tags to format and emphasize text content. These tags affect visual styling and may also convey meaning for screen readers and search engines.

1. Bold Text:

- <b> – Makes text bold, but without semantic importance.
- <strong> – Bold text with semantic emphasis (important content).

```
<p>This is <b>bold</b> text.</p>
<p>This is <strong>important</strong> text.</p>
```

2. Italic Text:

- <i> – Italicizes text without semantic emphasis (e.g., technical terms).
- <em> – Emphasized text, typically shown in italics with semantic meaning.

```
<p>This is <i>italic</i> text.</p>
<p>This is <em>emphasized</em> text.</p>
```

3. Underlined and Highlighted Text:

- <u> – Underlines text (not commonly used in modern design).
- <mark> – Highlights text with a yellow background.

```
<p>This is <u>underlined</u> text.</p>
<p>This is <mark>highlighted</mark> text.</p>
```

4. Small Text:

- <small> – Displays text in a smaller font size.

```
<p>This is <small>fine print</small>.</p>
```

5. Deleted and Inserted Text:

- `<del>` – Indicates deleted (strikethrough) text.
- `<ins>` – Indicates inserted (underlined) text.

```html
<p>This part was <del>removed</del> and this was <ins>added</ins>.</p>
```

Summary Table:

Tag	Description	Visual Effect
`<b>`	Bold text (no emphasis)	Bold
`<strong>`	Important/emphasized text	Bold
`<i>`	Italic (no emphasis)	Italic
`<em>`	Emphasized text	Italic
`<u>`	Underlined text	Underline
`<mark>`	Highlighted text	Yellow background
`<small>`	Smaller font size	Reduced text size
`<del>`	Deleted content	Strikethrough
`<ins>`	Inserted content	Underline

1.6 Hyperlinks: `<a>`

The `<a>` tag, known as the anchor tag, is used to create hyperlinks in HTML. It can link to external websites, internal pages, specific sections of the same page, email addresses, or files for download.

Key Attributes:

- `href` – Specifies the URL or path the link should navigate to.
- `target` – Defines how the link is opened (e.g., new tab or same window).
- `title` – Adds a tooltip shown when the mouse hovers over the link.
- `download` – Suggests the browser download the resource instead of navigating to it.

Example – Link to External Website:

```html
<a href="https://www.example.com" target="_blank">Visit Example</a>
```

- `target="_blank"` opens the link in a new tab.

Example – Link to an Internal Page:

```html
<a href="about.html">About Us</a>
```

Example – Link to a Section in the Same Page:

```html
<a href="#contact">Go to Contact Section</a>

<!-- Target Section -->
<h2 id="contact">Contact</h2>
```

Example – Downloadable File Link:

```html
<a href="files/report.pdf" download>Download Report</a>
```

Example – Email Link:

```
<a href="mailto:info@example.com">Email Us</a>
```

Best Practices:

- Always include meaningful link text (avoid “click here”).
- Use `target="_blank"` with caution—consider adding `rel="noopener noreferrer"` for security.
- Use `title` to provide additional context.
- Test links regularly to avoid broken references.

The anchor tag is essential for navigation and linking, forming the foundation of interconnected web pages.

1.7 Image Embedding: `<img>`

The `<img>` tag is used to embed images. It is a self-closing tag and does not wrap any content.

Important Attributes:

- `src` – URL or path to the image file.
- `alt` – Alternative text for screen readers and fallback.
- `width`, `height` – Size in pixels or percentage.

Example:

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

The `alt` attribute is necessary for accessibility.

1.8 Video Embedding: `<video>`, `<source>`

The `<video>` element is used to embed media content with playback controls. `<source>` tags specify video file types for compatibility.

Example:

```
<video width="320" height="240" controls>
  <source src="sample.mp4" type="video/mp4">
  <source src="sample.ogg" type="video/ogg">
  Your browser does not support the video tag.
</video>
```

Always provide multiple `<source>` formats to support different browsers.

1.9 IFrame: `<iframe>`

Embeds external content like maps or videos from YouTube.

```
<iframe width="560" height="315"
  src="https://www.youtube.com/embed/dQw4w9WgXcQ"
  frameborder="0" allowfullscreen>
</iframe>
```

1.10 Lists in HTML: `<ul>`, `<ol>`, `<li>`

HTML provides two main types of lists to structure related items: unordered and ordered lists. Each list is composed of one or more list items defined using the `<li>` tag.

- `<ul>` – Creates an unordered list with bullet points.
- `<ol>` – Creates an ordered list with numbered items.
- `<li>` – Defines a list item inside either `<ul>` or `<ol>`.

Lists can also be nested to create sub-lists, which is helpful for representing hierarchical data.

Example – Unordered List:

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
```

Example – Ordered List:

```
<ol>
  <li>Install VS Code</li>
  <li>Write HTML</li>
  <li>Preview in Browser</li>
</ol>
```

Example – Nested Lists:

```
<ul>
  <li>Frontend
    <ul>
      <li>HTML</li>
      <li>CSS</li>
    </ul>
  </li>
  <li>Backend
    <ol>
      <li>Node.js</li>
      <li>PHP</li>
    </ol>
  </li>
</ul>
```

List elements help organize content clearly and improve both readability and accessibility. Proper semantic usage of lists also enhances the screen reader compatibility.

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

HTML tables are used to organize data in rows and columns. They are structured using a combination of elements:

- `<table>` – Wraps the entire table.
- `<tr>` – Defines a table row.
- `<th>` – Header cell (bold and centered by default).
- `<td>` – Standard data cell.

Example:

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Alice</td>
    <td>24</td>
  </tr>
  <tr>
    <td>Bob</td>
    <td>30</td>
  </tr>
</table>
```

Additional elements include `<thead>`, `<tbody>`, `<tfoot>`, and attributes like `colspan` and `rowspan` for more complex layouts.

1.12 Forms and Input Fields

Used to collect data. `method="get"` appends data to URL; `post` sends data in request body. `action` defines the server endpoint.

```
<form action="submit.php" method="post">
  <label>Name: <input type="text" name="name"></label><br>
  <label>Message:<br>
    <textarea name="message"></textarea>
  </label><br>
  <input type="submit" value="Send">
</form>
```

1.12.1 Form Method Attribute: `method`

The `method` attribute of the `<form>` tag specifies how form data should be sent to the server. It has two common values: `get` and `post`.

1. `method="get"`

- Appends form data to the URL in name/value pairs.
- Visible in the browser's address bar.
- Data length is limited by the URL length.
- Suitable for non-sensitive data, like search queries or filters.

Example – GET Method:

```html
<form action="/search" method="get">
  <label>Search: <input type="text" name="q"></label>
  <button type="submit">Go</button>
</form>
```

Resulting URL:

`/search?q=example`

2. `method="post"`

- Sends form data in the HTTP request body.
- Not visible in the URL.
- No size limitations on the data (compared to GET).
- Preferred for sensitive or large amounts of data (e.g., passwords, messages).

Example – POST Method:

```html
<form action="/submit" method="post">
  <label>Email: <input type="email" name="email"></label><br>
  <label>Password: <input type="password" name="password"></label><br>
  <button type="submit">Login</button>
</form>
```

Summary:

- Use `get` for read-only requests or bookmarking.
- Use `post` for sending private or complex data.

1.13 Buttons: `<button>`

The `<button>` tag defines a clickable button, which can be used to submit forms, reset form inputs, or trigger JavaScript actions. Unlike `<input type="submit">`, the `<button>` tag allows for richer content inside, including text, images, and HTML elements.

Button Types (type attribute):

- `type="submit"` – Submits the form data to the server (default behavior).
- `type="reset"` – Resets all form fields to their initial values.
- `type="button"` – Performs no default action; used with JavaScript.

Example – Submit Button:

```
<form action="/submit" method="post">
  <label>Username: <input type="text" name="user"></label><br>
  <button type="submit">Login</button>
</form>
```

Example – Reset Button:

```
<form>
  <input type="text" name="name">
  <button type="reset">Clear</button>
</form>
```

Example – JavaScript Trigger:

```
<button type="button" onclick="alert('Clicked!')">Click Me</button>
```

Advantages over `<input type="submit">`:

- Supports nested elements (e.g., icons or spans).
- Easier to style with CSS.
- More semantic and flexible.

The `<button>` tag is essential for interactive and accessible form interfaces.

1.14 HTML Comments

Used to add notes in HTML code that are not rendered by the browser.

```
<!-- This is a comment -->
```

2 CSS Syntax and Selectors

2.1 CSS Syntax

CSS uses selectors and declaration blocks.

```
selector {
  property: value;
}
```

2.2 Linking CSS in HTML: `<link>` Tag

To apply external stylesheets to an HTML document, the `<link>` tag is used within the `<head>` section. This allows for separation of content (HTML) and presentation (CSS), promoting better structure and reusability.

Syntax:

```
<link rel="stylesheet" href="css/style.css">
```

Attributes:

- `rel="stylesheet"` – Declares the relationship as a stylesheet.
- `href` – Path to the external CSS file.
- `type="text/css"` – (Optional) MIME type of the stylesheet.

Example – Full HTML with Linked CSS:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Styled Page</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <h1 class="title">Hello, World!</h1>
</body>
</html>
```

Folder Structure Example:

```
project/
 index.html
 css/
     style.css
```

CSS Example (style.css):

```
.title {
  color: blue;
  font-family: Arial, sans-serif;
}
```

Linking an external CSS file ensures consistency across multiple HTML pages and simplifies maintenance.

2.3 CSS Class vs ID: `.class` and `#id`

In CSS, both classes and IDs are used as selectors to apply styles to HTML elements. However, they differ in purpose, syntax, and behavior.

1. Class Selector (`.class`):

- Denoted by a period (`.`) prefix.
- Can be applied to **multiple elements**.
- Promotes reusability and consistent styling.

Example – Class:

```
<p class="highlight">This is highlighted text.</p>
<p class="highlight">This is also highlighted.</p>
```

```
.highlight {
  background-color: yellow;
  font-weight: bold;
}
```

2. ID Selector (`#id`):

- Denoted by a hash (`#`) prefix.
- Must be **unique per page**—only used for a single element.
- Often used for JavaScript hooks or anchors.

Example – ID:

```
<div id="main-section">Main content goes here.</div>
```

```
#main-section {
  padding: 20px;
  background-color: #f0f0f0;
}
```

Key Differences:

Feature	**Class (`.class`)**	**ID (`#id`)**
Prefix	`.  (Dot)`	`# (Hash)`
Usage Count	Multiple elements	One element (unique)
Specificity	Lower	Higher
Use Case	Reusable styles	Unique styles or anchors

Best Practices:

- Use classes for general styling.
- Use IDs for unique page elements or JavaScript references.
- Avoid styling heavily with IDs due to high specificity, which can complicate overrides.

2.4 Tag Selectors

Applies to all elements of that tag.

```
p {
  font-size: 16px;
  color: #333;
}
```

2.5 Hover and Media Queries

`:hover` applies styles on mouseover. Media queries allow responsive design.

```
a:hover {
  color: red;
}

@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

2.6 External CSS Linking

Connects a CSS file to an HTML file using `<link>` in the `<head>`.

```
<link rel="stylesheet" href="css/style.css">
```

3 Folder Structure

Organizes project files for maintainability.

```
project-folder/
 index.html
 css/
    style.css
 js/
    script.js
 images/
     banner.jpg
```

4 Basic VS Code Help

Visual Studio Code (VS Code) is a lightweight, powerful code editor ideal for web development. Below are some essential features, shortcuts, and recommended extensions to improve your HTML/CSS workflow.

4.1 Essential Shortcuts and Features

- **Reveal in File Explorer:**

- Right-click the file tab or file name in the sidebar.
  - Select `Reveal in File Explorer` (Windows) or `Reveal in Finder` (macOS).
- **Zoom In / Zoom Out:**
  - `Ctrl + =` or `Ctrl + Mouse Wheel Up` – Zoom In.
  - `Ctrl + -` or `Ctrl + Mouse Wheel Down` – Zoom Out.
  - `Ctrl + 0` – Reset Zoom.
- **Open Integrated Terminal:** `` Ctrl + ` `` (backtick)
- **Toggle Sidebar:** `Ctrl + B`
- **Quick File Open:** `Ctrl + P`
- **Go to Line:** `Ctrl + G`
- **Command Palette:** `Ctrl + Shift + P` – Access all commands and settings quickly.

4.2 Recommended Extensions for HTML & CSS

- **Live Server** – Launch a local development server with live reloading.
  - Author: Ritwick Dey
  - Shortcut: Right-click `index.html` → `Open with Live Server`
- **Prettier – Code Formatter** – Automatically format HTML, CSS, and JavaScript code.
- **HTML CSS Support** – Provides CSS class and ID autocomplete while writing HTML.
- **Auto Rename Tag** – Automatically renames matching HTML tag when you edit one.
- **Emmet** (built-in) – Enables abbreviation-based coding (e.g., `div.container>ul>li*3`)
- **Color Highlight** – Highlights CSS color codes in the editor.
- **Path Intellisense** – Autocompletes file paths while typing.

4.3 Tips

- Use the command palette (`Ctrl + Shift + P`) to search and run commands quickly.
- Enable autosave from `File → Auto Save` for smoother workflow.
- Use snippets to speed up writing repetitive code.

VS Code's flexibility and plugin ecosystem make it an excellent choice for learning and building web applications.