



# Milestone-1 Cheatsheet

---

## ◆ VS Code & What is HTML

- HTML stands for *HyperText Markup Language*.
  - Used to **structure content** on the web (text, links, images, etc).
  - **VS Code**: Popular code editor. Use ! + Tab for HTML boilerplate.
- 

## ◆ HTML Text — Paragraphs & Text Formatting

Tag	Description
<p>	Paragraph
<b>	Bold (no importance)
<i>	Italic (no emphasis)
<strong>	Important and bold
<em>	Emphasized and italic

### Example:

```
<p>This is <strong>important</strong> and <em>emphasized</em>.</p>
```

---

## ◆ Headings, <small>, Block vs Inline

Tag	Use
<h1> to <h6>	Headings (h1 is largest)
<small>	Smaller font text

<code>&lt;div&gt;</code>	Block-level container
<code>&lt;span&gt;</code>	Inline container

 **Block vs Inline:**

- **Block:** Starts on a new line, takes full width (`<div>`, `<p>`, `<h1>`)
  - **Inline:** Flows within text (`<span>`, `<b>`, `<i>`)
- 

◆ **Lists, `<br>`, `<button>`**

Tag	Description
<code>&lt;ol&gt;</code>	Ordered list (numbered)
<code>&lt;ul&gt;</code>	Unordered list (bulleted)
<code>&lt;li&gt;</code>	List item
<code>&lt;br&gt;</code>	Line break
<code>&lt;button&gt;</code>	Clickable button

 **Example:**

```
<ul>
  <li>Item 1</li>
  <li>Item 2<br>Line break here</li>
</ul>
<button>Click Me</button>
```

---

◆ **Links with `<a>`**

Attribute	Description
<code>href</code>	Destination URL

target=" _blank"	Opens in new tab
------------------	------------------

 **Example:**

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

---

◆ **Images with <img>**

Attribute	Description
src	Image source (URL or path)
alt	Description if image doesn't load

 **Examples:**

```


```

---

◆ **Forms & Inputs**

Tag/Attribute	Description
<form>	Form container
<input type="text">	Text input
<input type="password">	Password field
<select><option>	Dropdown menu
<button type="submit">	Submit form

 **Example (Login Form):**

```
<form>
  <input type="text" placeholder="Username">
  <input type="password" placeholder="Password">
```

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

---

## ◆ Basic HTML Structure

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Page Title</title>
  </head>
  <body>
    <!-- Your visible content here -->
  </body>
</html>
```

- `<!DOCTYPE html>` = Declares HTML5.
  - `<html>` = Root of HTML page.
  - `<head>` = Metadata, title, external links.
  - `<meta>` = Charset and other info.
  - `<title>` = Title on browser tab.
  - `<body>` = Page content (text, images, etc).
- 

## Learn and Explore CSS

---

## ◆ Introduction to CSS & Types

CSS (**Cascading Style Sheets**) is used to **style** HTML elements.

**Types of CSS:**

Type	How to Use
Inline	<code>&lt;h1 style="color: red;"&gt;Hello&lt;/h1&gt;</code>
Internal	Inside <code>&lt;style&gt;</code> in <code>&lt;head&gt;</code> of HTML
External	Link <code>.css</code> file using <code>&lt;link rel="stylesheet" href="style.css"&gt;</code>

---

#### ◆ CSS Text Styling & Units

Property	Example
<code>color</code>	<code>color: blue;</code>
<code>font-size</code>	<code>font-size: 16px;</code>
<code>text-align</code>	<code>text-align: center;</code>
<code>font-weight</code>	<code>font-weight: bold;</code>

#### Measuring Units:

- `px` = Fixed pixels
  - `em` = Relative to parent
  - `%` = Percentage of parent
  - `rem` = Root em (relative to root font-size)
- 

#### ◆ Basic Selectors

Selector	Example	Selects
Type	<code>h1 {}</code>	All <code>&lt;h1&gt;</code> elements
Class	<code>.box {}</code>	All elements with <code>class="box"</code>

ID	<code>#header {}</code>	Element with <code>id="header"</code>
Universal	<code>* {}</code>	All elements
Grouping	<code>h1, p {}</code>	All <code>&lt;h1&gt;</code> and <code>&lt;p&gt;</code> elements
Attribute	<code>input[type="text"] {}</code>	Text input fields

### Class vs ID:

- **Class:** reusable across elements (.)
  - **ID:** unique per page (#)
- 

### ◆ Mixed Selectors

Selector	Description
<code>.btn.primary</code>	Element with both <code>btn</code> and <code>primary</code> class
<code>div span</code>	All <code>&lt;span&gt;</code> inside a <code>&lt;div&gt;</code>
<code>a[href^="https"]</code>	Attribute selector (starts with)
<code>p &gt; span</code>	Direct child <code>span</code> in <code>p</code>

---

### ◆ CSS Sizing & Spacing

Property	Description
<code>margin</code>	Space <b>outside</b> the element
<code>padding</code>	Space <b>inside</b> the element
<code>border</code>	Line around element
<code>border-radius</code>	Rounded corners

 **Margin Shorthand:**

```
margin: 10px 20px 30px 40px; /* top right bottom left */
```

---

**◆ CSS Box Model**

The **Box Model** = margin → border → padding → content

Part	Use
padding	Inside space
border	Outer line
margin	Space outside
height/width	Size of content box

 Example:

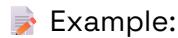
```
.box {  
  padding: 20px;  
  border: 1px solid #000;  
  margin: 10px;  
  width: 200px;  
}
```

---

**◆ CSS Display & Visibility**

Value	Description
block	Full width, new line
inline	Inside line, no width/height
inline-block	Like inline + can set size
none	Hides the element completely

<code>visibility: hidden</code>	Hides but keeps the space
<code>box-shadow</code>	Adds shadow around box



`box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2);`

---

#### ◆ CSS Backgrounds

Property	Example
<code>background-color</code>	<code>background-color: yellow;</code>
<code>background-image</code>	<code>background-image: url("bg.jpg");</code>
<code>background-repeat</code>	<code>no-repeat, repeat-x, repeat-y</code>
<code>background-size</code>	<code>cover, contain, 100% 100%</code>
<code>background-position</code>	<code>center, top right, 0 0</code>



```
body {
  background-image: url("img/bg.jpg");
  background-size: cover;
  background-position: center;
}
```

---



## Git, Source Control, GitHub, and Hosting

---

#### ◆ What is Version Control?

- **Version Control System (VCS)** = Tracks changes in code over time.
- **Git** = Local version control system.
- **GitHub** = Cloud hosting for Git repositories.

Term	Description
Git	CLI tool to track and manage code versions
GitHub	Platform to store Git repositories online

---

#### ◆ **Create GitHub Repository**

1. Go to <https://github.com>
2. Click "New Repository"
3. Name the repo → (e.g., `my-portfolio`)
4. Click **Create repository**

→ Copy repo link (HTTPS or SSH)

---

#### ◆ **Basic Git Commands**

Command	Description
<code>git init</code>	Initialize local Git repo
<code>git status</code>	Show current state
<code>git add .</code>	Stage all changes
<code>git add filename.ext</code>	Stage specific file

<code>git commit -m "message"</code>	Commit with a message
--------------------------------------	-----------------------

 Example:

```
git init  
git add .  
git commit -m "Initial commit"
```

---

### ◆ GitHub Account Setup

```
git config --global user.name "Your Name"  
git config --global user.email "your@email.com"
```

 Check Git version:

```
git --version
```

---

### ◆ Send Code to GitHub

1. Link local repo to remote:

```
git remote add origin https://github.com/username/repo.git
```

2. Push code:

```
git push -u origin main
```

 For existing GitHub repos:

```
git clone https://github.com/username/repo.git
```

---

### ◆ Common Issues for Beginners

Issue	Solution
Wrong branch name	Rename to <code>main</code> or push as new branch
Permission denied	Use HTTPS or SSH keys properly
Not staged	Run <code>git add .</code> before commit
Commit not pushed	Use <code>git push</code>

🛠 Check remote:

```
git remote -v
```

---

## ◆ Git Workflow Summary

✓ Basic Workflow:

```
# Step 1: Initialize
git init

# Step 2: Track changes
git add .

# Step 3: Save changes
git commit -m "Your message"

# Step 4: Connect to GitHub
git remote add origin <repo link>

# Step 5: Push to GitHub
git push -u origin main
```

---

## 🧩 More About HTML & CSS: Media, Layouts, Forms, and Flex Design

---

## ◆ HTML5 vs HTML & Embedding Media

### HTML5 Improvements:

- Semantic tags (`header`, `footer`, etc.)
- Supports audio/video tags
- Cleaner syntax, no need for type in script/link

### Audio Tag:

```
<audio controls>
  <source src="audio.mp3" type="audio/mpeg">
</audio>
```

### Video Tag:

```
<video width="320" controls>
  <source src="video.mp4" type="video/mp4">
</video>
```

### YouTube Embed:

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

---

## ◆ Semantic HTML5 Tags

Tag	Purpose
<code>&lt;nav&gt;</code>	Navigation menus
<code>&lt;main&gt;</code>	Main content

<code>&lt;header&gt;</code>	Page or section header
<code>&lt;footer&gt;</code>	Page footer
<code>&lt;section&gt;</code>	Group content by topic
<code>&lt;article&gt;</code>	Independent content block
<code>&lt;time&gt;</code>	Represent date/time

 Example:

```
<article>
  <header><h2>Blog Title</h2></header>
  <p>Post content...</p>
  <footer><time>2025-05-27</time></footer>
</article>
```

---

## ◆ HTML Forms (Advanced Elements)

Element	Use
<code>&lt;label&gt;</code>	Label for input
<code>&lt;fieldset&gt;</code>	Groups related form elements
<code>&lt;legend&gt;</code>	Title for fieldset
<code>&lt;textarea&gt;</code>	Multi-line input
<code>&lt;input type="radio"&gt;</code>	Single selection options
<code>&lt;input type="checkbox"&gt;</code>	Multiple options
<code>&lt;input type="reset"&gt;</code>	Clears all fields
<code>&lt;input type="submit"&gt;</code>	Submit form

 Example:

```
<form>
  <fieldset>
```

```

<legend>Survey</legend>
<label>Name: <input type="text" name="name"></label><br>
<label><input type="radio" name="gender" value="M"> Male</label>
<label><input type="radio" name="gender" value="F">
Female</label><br>
<textarea rows="4" cols="30"></textarea>
<input type="submit">
</fieldset>
</form>

```

---

## ◆ HTML Tables

Tag	Description
<table>	Start table
<tr>	Table row
<td>	Table cell
<th>	Table heading
<caption>	Title of table
<thead>	Header group
<tbody>	Body group
<tfoot>	Footer group

### Example:

```

<table>
  <caption>Student Marks</caption>
  <thead><tr><th>Name</th><th>Score</th></tr></thead>
  <tbody><tr><td>Ali</td><td>95</td></tr></tbody>
  <tfoot><tr><td>Total</td><td>100</td></tr></tfoot>
</table>

```

---

## ◆ CSS Flex Layout Overview

**Flexbox** = Powerful layout system for 1D alignment

Property	Value Example
display	flex
flex-direction	row, column
justify-content	center, space-between
align-items	center, stretch
gap	gap: 20px;

 Example:

```
.container {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
}
```

---

## ◆ Box Model Recap + auto

**Box model** = content + padding + border + margin

 **margin: auto;** → centers block-level elements

```
.container {  
  width: 500px;  
  margin: 0 auto; /* center horizontally */  
}
```

---

## ◆ Navigation & Internal Linking

```
<nav>  
  <a href="index.html">Home</a>  
  <a href="about.html">About</a>  
</nav>
```

## Internal Navigation (Same Page):

```
<a href="#section1">Go to Section 1</a>
<section id="section1">...</section>
```

---

## ◆ Hero Section with Flexbox

```
<section class="hero">
  <div class="hero-text">
    <h1>Welcome</h1>
    <p>Your site starts here.</p>
  </div>
  
</section>
```

```
.hero {
  display: flex;
  align-items: center;
  justify-content: space-between;
  background: #f0f0f0;
  padding: 50px;
}
```

---

## Pseudo Classes

Pseudo-classes select elements based on state, position, or interaction.

### ◆ :hover

- Trigger styles when the mouse is over the element.

```
button:hover {
  background-color: blue;
  color: white;
```

```
}
```

## ◆ :focus

- Applies when an input element is focused (clicked/selected).

```
input:focus {  
    border: 2px solid green;  
}
```

## ◆ :visited

- Styles links **after** they've been clicked.

```
a:visited {  
    color: purple;  
}
```

---

# ⌚ Advanced Pseudo-classes & Pseudo-elements

## ◆ :first-child / :nth-child(n)

- Target elements based on order in the parent.

```
li:first-child {  
    color: red;
```

```
}
```

```
li:nth-child(3) {  
    font-weight: bold;  
}
```

## ◆ :before and :after (Pseudo-elements)

- Add content before/after an element — useful for decorations, icons, labels.

```
h1::before {  
    content: "🔥 ";  
}
```

```
h1::after {  
    content: "💡 ";  
}
```

💡 Note: You need **display: block/inline-block** when using pseudo-elements if you apply box styling.

---

## ⌚ CSS Positioning

Positioning controls how elements are laid out in relation to others or the browser window.

### ◆ **static** (default)

- Normal flow, no special positioning.

```
div {  
  
    position: static;  
  
}
```

### ◆ **relative**

- Position relative to itself (you can shift it with **top**, **left**, etc.)

```
.box {  
  
    position: relative;  
  
    top: 10px; /* moves down */  
  
    left: 20px; /* moves right */  
  
}
```

### ◆ **absolute**

- Positioned relative to **nearest positioned ancestor** (not **static**)

```
.parent {  
  
    position: relative;  
  
}  
  
.child {  
  
    position: absolute;
```

```
    top: 0;  
  
    right: 0;  
  
}
```

## ◆ **fixed**

- Positioned relative to **browser window** (never moves on scroll)

```
.sticky-header {  
  
    position: fixed;  
  
    top: 0;  
  
    width: 100%;  
  
}
```

## ◆ **sticky**

- Acts like relative **until** a certain scroll point, then sticks like fixed.

```
nav {  
  
    position: sticky;  
  
    top: 0;  
  
}
```

---

## ◆ **z-index (Layering elements)**

- Controls the **stack order** of overlapping elements.
- Higher **z-index** means closer to the user.

```
.box1 {
  position: absolute;
  z-index: 1;
}

.box2 {
  position: absolute;
  z-index: 10; /* Will be on top */
}
```

---

## Bonus Tips:

Concept	Real Use Case Example
<code>:hover</code>	Button or link animation on mouseover
<code>:focus</code>	Highlight active input fields
<code>::before</code>	Add icons before text using <code>content</code>
<code>position: fixed</code>	Floating navbar, sticky buttons

<b>z-index</b>	Dropdowns, modals over other content
----------------	--------------------------------------

---