HTML & CSS Lab Manual – CSCL 1108
CSCL 1108 - Introduction to Computer Science Lab

Program: BS Computer Science (BSCS)
Semester: 1st Semester
Instructor: Almeerah Saleem
Academic Year: Fall 2025

General Notes

- Replace all 251XXXX with your actual Registration ID.
- Use **VS Code** for all exercises.
- Save each file in a clearly named folder (for example, html_lab_251XXXX).
- All work must be **pushed to your GitHub repository** after completion.
- Anything inside <!-- --> is a **comment** in HTML (used to explain code).
- Type your code manually do not copy-paste from online examples.
- Use **Live Preview** or **Open with Browser** to test your pages.
- AI tools are not allowed in any project, lab task, or exam.

Submission

- Each lab part should be committed and pushed to your GitHub repository under its own folder (e.g., part1_html_basics, part2_text_tags, etc.).
- Each commit must include proper message.
- The final HTML + CSS project will also be uploaded to GitHub before mid exam.

Part 1: Introduction to HTML

Concept

HTML (HyperText Markup Language) is the foundation of every web page. It structures content using **tags**, which tell the browser how to display information.
Every HTML document has a **basic structure** a head (information about the page) and a body (the visible part).

HTML Page Structure

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Web Page</title>
</head>
<body>
  <h1>Hello, World!</h1>
  <p>This is my very first web page.</p>
</body>
</html>
```

Explanation:

- <!DOCTYPE html> → Tells the browser this is an HTML5 document.
- <html> → Root tag containing the entire page.
- <head> → Holds meta info, title, and links to styles/scripts.
- <body> → Displays all visible content.
- <h1> → Main heading.
- <p> → Paragraph.

Lab Task

- Open VS Code and create a new file named:
  intro_251XXXX.html
- Type the above code manually.
- Change the <title> to your full name and Reg ID.
- Modify the <h1> to say:
  "Welcome to My Web Page – 251XXXX"
- Add another paragraph about yourself.
- Save and open the page in your browser.

Task

- Add another heading <h2> that says "My Learning Goals."
- Add a short paragraph about what you hope to learn in this course.
- Try adding a line break (<br>) or horizontal line (<hr>).

Part 2: Working with Text in HTML

Concept

Most web pages are made of **text content**, organized using headings, paragraphs, and lists.
HTML gives us different tags to represent structure and meaning, not just appearance.

Common Text Tags

Tag	Description	Example
<h1> to <h6>	Headings (h1 is largest)	My Page Title
<p>	Paragraph	This is a paragraph.
 or 	Bold text	Important
<i> or 	Italicized text	<i>Note this</i>
<u>	Underlined text	<u>Highlighted</u>
 	Line break	First line Second line
<hr>	Horizontal line	<hr>
	Unordered list (bullets)	Item
	Ordered list (numbers)	Step 1
	List item	Milk

Semantic Text Tags

Semantic tags describe *meaning* rather than just appearance.

Tag	Meaning	Example
	Important text	Warning!
	Emphasized text	Note carefully.
<mark>	Highlighted text	<mark>Key point</mark>
<small>	Smaller text	© 2025
<blockquote>	Quoted text	<blockquote>Famous quote...</blockquote>

Lab Task

Create a new file named **texttags_251XXXX.html**

Add a proper HTML structure (<!DOCTYPE html>, <html>, <head>, <body>).

Inside <body>, add the following:

- A main heading (<h1>My First HTML Text Page</h1>)
- A paragraph about your favorite subject.
- A subheading (<h2>Things I Like</h2>) followed by an unordered list of at least 3 items.
- Another subheading (<h2>Daily Routine</h2>) followed by an ordered list of 3–5 steps.
- Add one **horizontal line (**<hr>**)** between sections.
- Experiment with **bold**, **italic**, and **underline** inside one paragraph.
- Save and open in your browser.

Task

- Add a <blockquote> with a quote you like.
- Add <mark> around a word in your paragraph to highlight it.
- Add a <small> line at the bottom that says:
  "Created by 251XXXX"

Part 3: Hyperlinks and Images

Concept

Hyperlinks connect one page to another they’re the backbone of the web.
Images, meanwhile, make content visual and engaging.
HTML provides tags to easily add both.

Hyperlinks

The hyperlink tag is <a> (anchor).
It requires the href attribute to specify the destination.

```
<a href="https://www.w3schools.com">Visit W3Schools</a>
```

Attribute	Description	Example
href	Destination URL	GitHub
target="_blank"	Opens link in new tab	SZABIST
mailto:	Opens email app	Email Me
title	Tooltip text	Hover me

Images

The image tag is <img> it does not need a closing tag.

```
<img src="profile.jpg" alt="My Profile Photo" width="200" height="200">
```

Attribute	Description	Example
src	File path or URL of image	src="images/photo.png"
alt	Alternate text (for screen readers or errors)	alt="Profile picture"
width, height	Set size of image	width="250"
title	Shows text on hover	title="My Photo"

Tip:
If your image is in a subfolder, use a **relative path**:

```
<img src="images/mypic.png" alt="Profile">
```

If you use an online placeholder:

```
<img src="https://placehold.co/200x200" alt="Placeholder Image">
```

Lab Task

- Create a file named **links_251XXXX.html**.
- Add a proper HTML structure.
- Inside <body>, include the following:
- Heading: “My Links and Images”
- Paragraph describing your interests.
- Add **three hyperlinks**:
  - One to your favorite website.
  - One to mailto:your_email@example.com
  - One to your GitHub profile (use your actual GitHub link).
- Add one **image** (either a local file or a placeholder URL).
- Add an **alt attribute** for accessibility.
- Add <hr> after the image.

Common Mistakes

Forgetting to close <a> tags.

Not using quotes around attribute values.
Using \ instead of / in image paths.
Forgetting the alt attribute (always required).

Task

Add a link that opens your GitHub profile **in a new tab**.
Make the image itself clickable clicking it should open your GitHub profile:

```
<a href="https://github.com/YourUsername" target="_blank">
  <img src="https://placehold.co/150x150" alt="My GitHub">
</a>
```

Add a tooltip (title) to one of your links.

Part 4: Tables and Forms in HTML

Concept

Tables are used to organize data in rows and columns, while forms collect input from users.
In modern web design, tables are used for **data presentation**, not layout but understanding them is essential.

Table Structure

A basic HTML table uses these main tags:

Tag	Purpose
<table>	Starts the table
<tr>	Table row
<th>	Table header cell (bold by default)
<td>	Table data cell
<caption>	Optional table title

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Table – 251XXXX</title>
</head>
<body>
  <h1>Student Information</h1>

  <table border="1" cellpadding="8" cellspacing="0">
    <caption>Class Details</caption>
    <tr>
      <th>Reg ID</th>
      <th>Name</th>
      <th>Email</th>
      <th>Program</th>
      <th>Section</th>
    </tr>
    <tr>
      <td>251XXXX</td>
      <td>Ali Khan</td>
      <td>alikhan@example.com</td>
      <td>BSCS</td>
      <td>A</td>
    </tr>
    <tr>
      <td>251XXXX</td>
      <td>Sara Ahmed</td>
      <td>sara@example.com</td>
      <td>BSCS</td>
      <td>A</td>
    </tr>
  </table>
</body>
</html>
```

Explanation

border adds visible borders (simple styling).
cellpadding adds inner spacing inside cells.
cellspacing adds space between cells.
<th> cells are automatically **bold** and **centered**.

Forms (Basic Introduction)

Forms are used to collect data.
They include elements like text boxes, dropdowns, checkboxes, etc.
Example:

```
<form action="#" method="post">
  <label for="name">Full Name:</label>
  <input type="text" id="name" name="name"><br><br>

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

  <input type="submit" value="Submit">
</form>
```

Tag	Description
<form>	Wraps input fields
<label>	Describes an input field
<input>	Accepts user input
<select>	Dropdown list
<textarea>	Multi-line input
<button>	Clickable button

Lab Task

Create a new file named **tables_251XXXX.html**.
Build a table with the following columns:
Reg ID | Name | Email | Program | Section | GitHub
Add your own details in one row.
Use border, cellpadding, and cellspacing attributes.
Below the table, create a simple form with:

- Full Name
- Email
- Program (dropdown with 3–4 options)
- Submit button

Save and open in browser.

Common Mistakes

Missing closing tags (</tr>, </td>, etc.).
Forgetting quotes around attribute values.
Placing <form> tags inside tables incorrectly.
Using forms without labels.

Task

- Add one more student row in your table.
- Use the <caption> tag to give your table a title.

- Add a new input field in your form for “Favorite Language.”
- Try making one input field **required**:

```
<input type="text" name="fullname" required>
```

Part 5: Introduction to CSS

Concept

CSS (**Cascading Style Sheets**) controls how HTML elements look their colors, fonts, borders, spacing, and layout.
While HTML gives *structure*, CSS gives *style*.
There are **three ways** to add CSS to a web page:

- **Inline CSS** – inside a tag (not recommended for full projects).
- **Internal CSS** – inside <style> tags in the <head>.
- **External CSS** – in a separate .css file (best practice).

---

1. Inline CSS

Directly add style to a tag.

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

Use only for quick testing, not full pages.

---

2. Internal CSS

Written between <style> tags inside the <head>.

```
<!DOCTYPE html>
<html>
<head>
<style>
  body {
    background-color: #f7f9ff;
    font-family: Arial, sans-serif;
  }
  h1 {
    color: #333333;
  }
</style>
</head>
<body>
  <h1>Welcome</h1>
  <p>This page uses internal CSS.</p>
</body>
</html>
```

---

3. External CSS

This is the **recommended** way.
All your CSS rules are written in a separate file (e.g., style.css).

HTML file:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Styled Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Hello CSS!</h1>
  <p>This page is styled externally.</p>
</body>
</html>
```

style.css:

```
body {
  background-color: #f7f9ff; /* HEX code for a light blue shade */
  font-family: Verdana, sans-serif;
```

```
}
h1 {
    color: #003366;
    text-align: center;
}
p {
    color: #555555;
    font-size: 16px;
}
```

Explanation of Key Elements

Property	Description	Example
color	Sets text color	color: blue;
background-color	Changes background	background-color: #f7f9ff;
font-family	Sets text font	font-family: Arial, sans-serif;
font-size	Adjusts text size	font-size: 18px;
text-align	Aligns text	text-align: center;
border	Adds border	border: 2px solid black;
margin	Space outside element	margin: 10px;
padding	Space inside element	padding: 10px;

Lab Task

Create a folder named **css_lab_251XXXX**.

Inside it, create two files:

    style_251XXXX.css
    css_intro_251XXXX.html

Link the `.css` file in your HTML `<head>` using the `<link>` tag.

Add styles to your CSS file:

- Change the background color of the page.
- Center the heading.
- Make paragraph text gray and increase its font size.
- Add padding to the body (20px).
- Add a border around the heading (`border: 2px solid black;`).

Save and open in your browser.

Common Mistakes

Forgetting to link the CSS file correctly (check folder path!).

Missing the ; at the end of each CSS rule.

Using wrong property names (CSS is case-sensitive).

Putting CSS code outside `<style>` in internal CSS.

Task

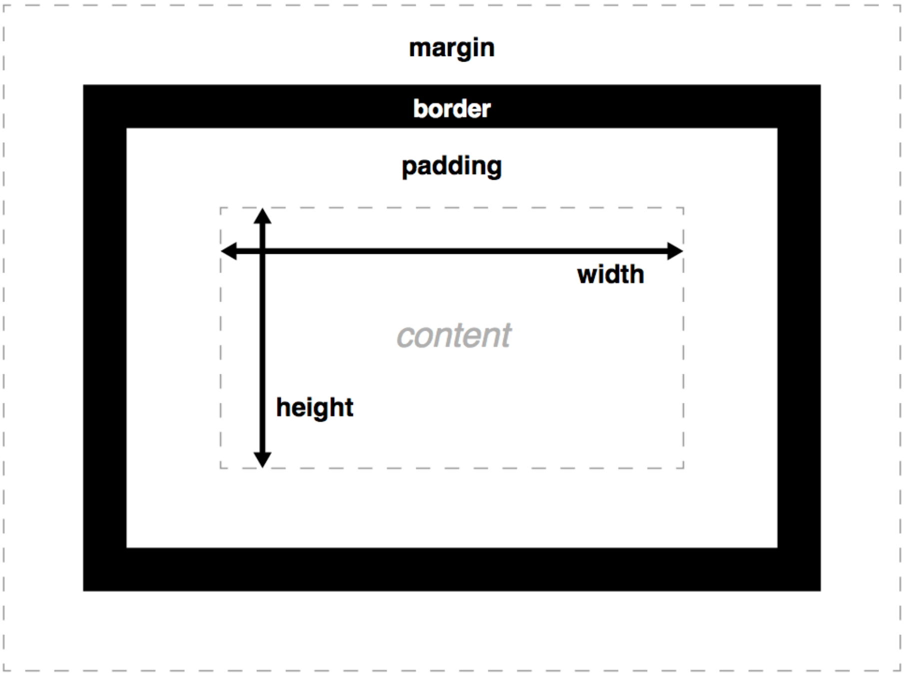
- Add one more CSS rule to style all `<h2>` elements with a color of your choice.
- Change the body font to a Google Font (like "Poppins" or "Open Sans").
- Experiment with different background colors using HEX codes (like `#fce4ec`, `#e3f2fd`, or `#f9fbe7`).

Part 6: CSS Formatting and Layout (The Box Model)

Concept

Every element in a web-page can be thought of as a **box**.
CSS lets you control the **size, spacing, borders, and background** of these boxes.
This concept is called the **Box Model**.
Each element's total space =
content + padding + border + margin

Box Model Structure



Common CSS Properties

Property	Description	Example
border	Adds a border	border: 2px solid black;
border-radius	Rounds corners	border-radius: 8px;
padding	Space inside element	padding: 15px;
margin	Space outside element	margin: 20px;
background-color	Sets background color	background-color: #f7f9ff;
width / height	Sets element size	width: 400px; height: 200px;
text-align	Aligns text inside box	text-align: center;
box-shadow	Adds soft shadow	box-shadow: 0 0 10px lightgray;

Example

HTML:

```
<!DOCTYPE html>
<html>
<head>
   <title>Box Model Demo – 251XXXX</title>
   <link rel="stylesheet" href="boxmodel_251XXXX.css">
</head>
<body>
   <h1>Welcome to My Page</h1>
```

```html
    <div class="info-box">
        <h2>About Me</h2>
        <p>Hello! I am learning HTML and CSS at SZABIST.</p>
    </div>
</body>
</html>
```

CSS:

```css
body {
    background-color: #f7f9ff; /* light blue */
    font-family: Arial, sans-serif;
    padding: 20px;
}

.info-box {
    background-color: white;
    border: 2px solid #003366;
    border-radius: 12px;
    padding: 20px;
    margin: 20px auto;
    width: 400px;
    box-shadow: 0 0 8px lightgray;
    text-align: center;
}
```

Explanation

The `.info-box` is a *div container* styled as a box.

`margin: 20px auto;` centers it horizontally.

`border-radius` softens edges.

`box-shadow` gives a clean 3D effect.

Lab Task

Create two files:

- boxmodel_251XXXX.html
- boxmodel_251XXXX.css

Build a simple info card (like the example).

Add a `<div>` with your name and Reg ID inside it.

Apply border, padding, background, and border-radius.

Center it on the page.

Add one more box below it with a small paragraph titled “My Goals.”

Save and open in browser.

Common Mistakes

Forgetting to add . before class names in CSS (e.g., `.info-box`).

Using commas instead of semicolons in CSS.

Confusing `margin` and `padding`.

Forgetting to link CSS file correctly.

Task

Add a subtle **box shadow** to both boxes.

Use `border-radius: 50%;` on a small image to make it circular.

Try `text-align: justify;` in your paragraph.
Use different background shades for alternating boxes.

Part 6.1: CSS Shorthand Properties & Effects

Concept

Some CSS properties can take **multiple values in a single line**.
This is called **shorthand syntax**, and it helps you write cleaner, shorter code.
It's especially common for properties that deal with spacing, borders, or effects.

---

1. Box Shadow (Recap)

Full syntax:

```
box-shadow: offset-x offset-y blur-radius spread-radius color;
```

You can control:

- Direction (x/y)
- Sharpness (blur)
- Size (spread)
- Color

Examples:

```
/* Single soft shadow */
box-shadow: 5px 5px 15px 0 rgba(0,0,0,0.2);

/* Multiple shadows */
box-shadow: 3px 3px 5px gray, -3px -3px 5px lightgray;
```

---

2. Margin & Padding

Margin and padding accept **1 to 4 values**:

Number of Values	Meaning	Example
1	All four sides are the same	margin: 10px;
2	Top–Bottom and Left–Right	margin: 10px 20px;
3	Top, Left–Right, Bottom	margin: 5px 10px 15px;
4	Top, Right, Bottom, Left	margin: 5px 10px 15px 20px;

Shortcut tip: Think clockwise → **Top** → **Right** → **Bottom** → **Left (TRBL)**

Example:

```
padding: 10px 15px;   /* vertical 10px, horizontal 15px */
margin: 20px;         /* same on all sides */
```

---

3. Border

Full syntax:

```
border: width style color;
```

Examples:

```
border: 2px solid black;
border: 1px dashed #999;
border: 3px double blue;
```

You can also set individual sides:

```
border-top: 2px solid red;
border-bottom: 1px dotted gray;
```

---

4. Font & Background

Font shorthand:

```
font: font-style font-weight font-size font-family;
```

Example:

```
font: italic bold 16px Arial;
```

Background shorthand:

```
background: color image repeat position;
```

Example:

```
background: #f7f9ff url('bg.png') no-repeat center;
```

---

5. Transition (for hover effects)

Full syntax:

```
transition: property duration timing-function delay;
```

Example:

```
transition: all 0.3s ease;
```

This means “animate any property change over 0.3 seconds smoothly.”

Quick Shortcut Summary

Property	Shorthand Example	Meaning
margin	margin: 10px 15px;	Top/Bottom 10px, Left/Right 15px
padding	padding: 5px 10px;	Vertical 5px, Horizontal 10px
border	border: 2px solid black;	All borders uniform
box-shadow	5px 5px 10px lightgray;	Offset shadow with blur
font	font: italic bold 16px Arial;	Style, weight, size, family
transition	transition: all 0.3s ease;	Smooth animation effect

---

Extra Notes

mailto:

In HTML, mailto: is used to make an email link that opens the user’s email app.

```
<a href="mailto:student@szabist.pk">Email Me</a>
```

<footer>

Used to display page footnotes or copyright info.

```
<footer>Created by 251XXXX – SZABIST</footer>
```

Tip:

Use small text or gray color for footers:

```
footer {
  text-align: center;
  font-size: 12px;
  color: gray;
  margin-top: 20px;
}
```

Part 7: Personal Webpage – Mini Project

Concept

Now that you know how to use **HTML structure** and **CSS styling**, it's time to create your own small webpage combining all major concepts.

Objective

Design a **personal webpage** that includes your name, a short introduction, contact links, and a styled layout using CSS.

Requirements

Files

- personalpage_251XXXX.html
- personalpage_251XXXX.css

Structure

Your webpage must include:

Header

- Contains your name and Reg ID.
- Styled with a background color.
- Text aligned to center.

About Me Section

- A short paragraph introducing yourself.
- Use one or more text formatting tags (<b>, <i>, <mark>).

Favorites Section

- A list of at least three favorite things (e.g., color, food, hobby).
- Each listed item styled differently (font color, size, or background).

Contact Section

- A simple link to your GitHub profile.
- An email link using mailto:.

Footer

- Centered text: Created by 251XXXX – SZABIST

CSS Styling

Use **at least five** of these properties:

- background-color
- color
- border
- border-radius
- font-family
- margin
- padding
- text-align
- box-shadow
- width / height

Lab Task Steps

- Create both files in VS Code.
- Link your CSS file inside the <head>.
- Add content section by section (Header → About → Favorites → Contact → Footer).
- Style each section using class selectors.
- Open your HTML file in a browser and verify alignment and spacing.

Task

- Add a light shadow or gradient to your header.
- Create a hover effect for your GitHub link (a:hover).
- Use border-radius: 50%; to make your profile image circular.
- Add a transition so hover effects animate smoothly.

Part 8: Final Styling Touches & Best Practices

Concept

Professional developers write HTML & CSS that is not only functional but **clean, consistent, and visually balanced**.
This part teaches small refinements that make their work look neat and submission-ready.

---

1. Organize Your Code

Good organization = easier debugging and better teamwork later on.

Tips:

Keep indentation consistent (use 2 or 4 spaces per level).
Always close your tags.
Use lowercase for all tag and attribute names.
Add meaningful comments.

Example:

```
<!-- Header section -->
<header>
  <h1>My Portfolio</h1>
</header>
```

In CSS:

```
/* Header styling */
header {
  background-color: #f7f9ff;
  padding: 15px;
}
```

---

2. Naming Conventions

Avoid random or generic names like `.box1` or `.div2`.
Use **semantic, descriptive names** so you can understand your code weeks later.

Example:

```
/* Poor naming */
.div1 { color: red; }

/* Better naming */
.header-title { color: red; }
```

---

3. Spacing & Alignment

Uniform spacing gives balance and professionalism.

Checklist:

Ensure elements aren't touching screen edges.
Use consistent `margin` and `padding`.
Align text with `text-align: left;` or `center;` where suitable.
For wider layouts, use `max-width` to prevent stretched content.

```
body {
  margin: 0 auto;
  max-width: 800px;
  font-family: "Poppins", sans-serif;
}
```

---

4. Color & Contrast

Use color to **highlight**, not overwhelm.
Keep text readable dark text on light background or vice versa.

Use consistent colors throughout.
Tip:
Use HEX values (#f7f9ff), RGB (rgb(230, 230, 250)), or HSL (hsl(240, 100%, 97%)) but pick one style and stick to it.

---

5. Fonts & Readability

Fonts influence how professional your webpage looks.
Tips:
Choose clean fonts like Arial, Verdana, Poppins, Roboto.
Limit font sizes: headings (18–24px), paragraphs (14–16px).
Use line-height to add breathing space between lines.

```
p {
  font-size: 16px;
  line-height: 1.6;
  text-align: justify;
}
```

---

6. Using Hover Effects (Optional Enhancement)

Adding hover effects gives interaction and life to your webpage.

```
a {
  color: #3366cc;
  text-decoration: none;
  transition: color 0.3s ease;
}

a:hover {
  color: #ff6600;
  text-decoration: underline;
}
```

---

7. Consistent Footer & Header Design

Uniform headers and footers make your page feel complete.

```
header, footer {
  background-color: #f1f1f1;
  text-align: center;
  padding: 10px;
  border-radius: 5px;
}
```

---

8. Validating and Testing Your Webpage

Before submitting, **always check**:

- No missing closing tags (</div>, </p>).
- CSS is properly linked (<link rel="stylesheet"> path is correct).
- The page works in multiple browsers (Chrome, Edge, Firefox).
- Links open correctly and colors appear as intended.

---

9. Task

Task:
Add a simple hover animation or border effect on your “Favorites” box.
Make it grow slightly or change color smoothly when hovered.
Example:

```
.favorites:hover {
  transform: scale(1.02);
```

```
    box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.2);
}
```

Hint: Use the `transition` property for smoothness.

---

10. Submission Reminder

Before you submit:

- Ensure your code is clean and readable.
- Folder and file names match the required format (e.g., part8_finaltouches_251XXXX).
- Commit and push your folder to GitHub with a clear message:

```
git add .
git commit -m "Added final styling touches and best practices"
git push origin main
```

VS Code Tips & Shortcuts

Basic Shortcuts

Action	Shortcut	Description
Open new file	Ctrl + N	Create a new blank file
Open file	Ctrl + O	Open an existing file
Save file	Ctrl + S	Save current file
Save all files	Ctrl + K → S	Save all open tabs
Open folder/project	Ctrl + K → O	Open your project folder
Split screen view	Ctrl + \	View multiple files side-by-side
Toggle sidebar	Ctrl + B	Show or hide the left explorer bar
Comment / Uncomment line	Ctrl + /	Toggle comments in code
Format document	Shift + Alt + F	Auto-indent and tidy up your code
Duplicate line	Shift + Alt + ↓	Copy current line down
Delete line	Ctrl + Shift + K	Delete current line quickly
Wrap tag in HTML	Alt + W	Wrap selected text in an HTML tag (with Emmet enabled)
Open Live Server	Alt + L → Alt + O	Run and preview your webpage live in browser

---

Tips

Type `!` **and press Tab or Enter** → VS Code auto-generates a full HTML5 boilerplate (works if Emmet is active).

To quickly link your CSS:

Type `link` and press **Tab** → VS Code inserts `<link rel="stylesheet" href="">`.

Right-click the HTML file → **"Open with Live Server"** to preview instantly.

If your color code doesn't show a preview square ensure the file type is **.css** and the line ends with a semicolon.

Useful Extensions for Beginners

Extension	Author	Purpose
Live Server	Ritwick Dey	Instantly opens your HTML file in a browser and refreshes automatically when you save changes.

Extension	Author	Purpose
Image Preview	Kiss Tamás	Shows a small thumbnail when you hover over an image file path helps verify correct image links.
Open Browser Preview	Eno Yao	Opens a live web preview directly *inside* VS Code useful when external browser access is limited.
Prettier – Code Formatter	Prettier	Automatically formats your code for consistent indentation and spacing.
HTML CSS Support	ecmel	Suggests and auto-completes CSS class names in your HTML files.
Color Highlight	Sergii N.	Displays color boxes next to HEX, RGB, or HSL codes makes styling more visual.

Quick Setup Tip

After installing extensions:
Restart VS Code once.
Go to **Settings → Extensions → Manage** to enable/disable individually as needed.
Optional: turn on **Auto Save** (File → Auto Save) for faster workflow.
