ISDM (INDEPENDENT SKILL DEVELOPMENT MISSION)

STUDY MATERIAL FOR WEEK 1: INTRODUCTION TO HTML

1. UNDERSTANDING HOW WEBSITES WORK

What is a Website?

A **website** is a collection of web pages that are accessed through the internet. These pages are built using **HTML (HyperText Markup Language)** and are often styled using **CSS (Cascading Style Sheets)** and made interactive using **JavaScript**.

How Websites Work?

1. **User Requests a Website** – When you enter a website URL (like www.google.com) in a browser, your computer sends a request to a server.
2. **Server Processes the Request** – The server stores website files and responds to the request by sending back the necessary HTML, CSS, and JavaScript files.
3. **Browser Renders the Website** – The browser reads the HTML file and displays the web page based on the provided content and styling.

Types of Websites

- **Static Websites** – Simple websites built using only HTML and CSS. Example: A personal portfolio.
- **Dynamic Websites** – Websites that change based on user interaction and use programming languages like JavaScript, PHP, or Python. Example: Facebook, YouTube.

---

2. Introduction to HTML and Its Importance

What is HTML?

HTML (HyperText Markup Language) is the standard language used to create and structure web pages. It provides the **basic framework** for a webpage, defining elements like text, headings, images, and links.

Importance of HTML

✅ **Foundation of the Web** – Every website is built using HTML.
✅ **Easy to Learn** – Beginners can quickly understand and start creating web pages.
✅ **Supports Multimedia** – HTML can display images, videos, and audio files.
✅ **SEO-Friendly** – Well-structured HTML helps in **search engine optimization (SEO)** for better visibility on search engines.

Basic HTML Example

```
<!DOCTYPE html>

<html>

<head>

  <title>My First Webpage</title>
```

```
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is a simple HTML page.</p>
</body>
</html>
```

---

3. Setting Up a Code Editor (VS Code, Sublime, or Notepad++)

To write HTML code, you need a **code editor**. Below are the most commonly used editors:

Recommended Code Editors

1. **VS Code (Visual Studio Code)**
   - Free and widely used for web development.
   - Supports extensions for better coding experience.
   - Download: https://code.visualstudio.com/
2. **Sublime Text**
   - Lightweight and fast.
   - Offers syntax highlighting and auto-completion.
   - Download: https://www.sublimetext.com/
3. **Notepad++**
   - Simple and best for beginners.

- Less heavy on system resources.
- Download: https://notepad-plus-plus.org/

Steps to Set Up VS Code

1. Download and install **VS Code** from the official website.
2. Open VS Code and create a new file with the extension **.html** (e.g., index.html).
3. Start writing your HTML code.
4. Open the file in a web browser to view your page.

---

4. Basic HTML Document Structure

Every HTML document follows a standard structure:

HTML Structure Explanation

```
<!DOCTYPE html>  <!-- Defines the document type -->
<html>  <!-- Root element of the webpage -->
<head>
  <title>Page Title</title>  <!-- Title shown on the browser tab -->
</head>
<body>
  <h1>Welcome to HTML</h1>  <!-- Main heading -->
  <p>This is a paragraph.</p>  <!-- Paragraph content -->
</body>
```

```
</html>
```

Key Elements of an HTML Page

- <html> – The root element of the webpage.
- <head> – Contains meta-information like title, styles, and links.
- <title> – Sets the title of the webpage.
- <body> – The main content of the page (text, images, links, etc.).

---

5. Creating Headings, Paragraphs, and Line Breaks

Headings (<h1> to <h6> Tags)

Headings are used to define titles on a webpage. HTML provides six levels of headings.

```
<h1>Main Heading</h1>

<h2>Subheading</h2>

<h3>Sub-subheading</h3>
```

Paragraphs (<p> Tag)

The <p> tag is used to define paragraphs in HTML.

```
<p>This is a paragraph of text.</p>
```

Line Breaks (
 Tag)

The
 tag is used to insert a line break without starting a new paragraph.

```
<p>This is a line.<br>This is the next line.</p>
```

---

6. Working with Lists (<ul>, <ol>, <li> Tags)

Unordered Lists (<ul>)

Creates a bullet-point list.

```
<ul>
  <li>Apple</li>
  <li>Banana</li>
  <li>Cherry</li>
</ul>
```

Ordered Lists (<ol>)

Creates a numbered list.

```
<ol>
  <li>Step 1: Open your code editor</li>
  <li>Step 2: Create an HTML file</li>
  <li>Step 3: Start writing your code</li>
</ol>
```

Nested Lists (Lists inside Lists)

```
<ul>
  <li>Fruits
    <ul>
```

```
        <li>Apple</li>
        <li>Banana</li>
      </ul>
    </li>
    <li>Vegetables</li>
</ul>
```

---

7. Assignment: Creating a Simple Webpage with Text and Lists

Task 1: Create a Basic HTML Page

1. Open your **code editor** (VS Code, Sublime, or Notepad++).
2. Create a new file and save it as index.html.
3. Write the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>My First HTML Page</title>
</head>
<body>
    <h1>Welcome to My Webpage</h1>
    <p>Hello! This is my first webpage created using HTML.</p>
```

```
  <h2>My Favorite Fruits</h2>
  <ul>
    <li>Apple</li>
    <li>Mango</li>
    <li>Banana</li>
  </ul>

  <h2>My Daily Routine</h2>
  <ol>
    <li>Wake up at 7 AM</li>
    <li>Exercise for 30 minutes</li>
    <li>Study and work</li>
    <li>Relax and sleep</li>
  </ol>
</body>
</html>
```

4. **Save the file** and open it in a web browser (Chrome, Edge, or Firefox).

Expected Output

The webpage will display:

- A **heading** with "Welcome to My Webpage".
- A **paragraph** introducing yourself.
- An **unordered list** of favorite fruits.
- An **ordered list** of a daily routine.

✅ **Submit your completed HTML file to the instructor.**

CONCLUSION

In this first week, you have learned: ✔ How websites work
✔ The importance of HTML
✔ Setting up a code editor
✔ Basic HTML structure
✔ Creating headings, paragraphs, and lists

🎯 **Next Week:** We will explore **adding images, hyperlinks, tables, and forms** in HTML. 🚀

Keep practicing, and happy coding! 🎉

WORKING WITH HTML ELEMENTS

1. ADDING IMAGES TO A WEBPAGE (<IMG> TAG)

What is the <img> tag?

The <img> tag is used to display images on a webpage. It requires the src attribute to specify the image source.

Basic Syntax

<img src="image.jpg" alt="Description of the image">

- src – Specifies the image location (can be a URL or local file path).
- alt – Provides an alternative text if the image doesn't load.

Example: Adding an Image

<img src="flower.jpg" alt="A beautiful flower" width="300" height="200">

- width and height define the image size.

Example: Displaying an Online Image

<img src="https://example.com/image.jpg" alt="Online Image">

---

2. CREATING HYPERLINKS (<A HREF> TAG)

What is a Hyperlink?

A hyperlink allows users to navigate from one webpage to another.

Basic Syntax

Visit Google

- href – Specifies the URL of the webpage.
- The text inside the <a> tag becomes the clickable link.

Opening Links in a New Tab

Open Google in New Tab

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

Linking to an Internal Page

About Us

- This links to another HTML page in the same project.

Adding an Email Link

Send an Email

---

3. Tables in HTML (<table>, <tr>, <td>, <th>)

Tables are used to display data in a structured format.

Basic Table Structure

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Age</th>
    <th>City</th>
```

```
  </tr>
  <tr>
    <td>John</td>
    <td>25</td>
    <td>New York</td>
  </tr>
  <tr>
    <td>Emma</td>
    <td>22</td>
    <td>Los Angeles</td>
  </tr>
</table>
```

Table Tags Explained

- <table> – Creates a table.
- <tr> – Table row.
- <th> – Table heading (bold and centered).
- <td> – Table data (cell content).

Adding a Table Caption

```
<table border="1">
  <caption>Student Information</caption>
  <tr>
```

```
    <th>Name</th>
    <th>Grade</th>
  </tr>
  <tr>
    <td>Alex</td>
    <td>A</td>
  </tr>
</table>
```

---

4. Forms and Input Fields (<form>, <input>, <button>, <textarea>)

Forms collect user input like names, emails, and messages.

Basic Form Example

```
<form action="submit.php" method="post">
  <label for="name">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>

  <label for="message">Message:</label><br>
```

```
<textarea id="message" name="message" rows="4"
cols="30"></textarea><br><br>


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

</form>
```

Common Form Elements

- <input type="text"> – Text field.
- <input type="email"> – Email input.
- <input type="password"> – Password field.
- <input type="radio"> – Radio buttons.
- <input type="checkbox"> – Checkboxes.
- <textarea> – Multiline text input.
- <button type="submit"> – Submit button.

Example: Login Form

```
<form>
  <label>Username:</label>
  <input type="text"><br><br>

  <label>Password:</label>
  <input type="password"><br><br>
```

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

---

5. Working with Semantic Elements (<section>, <article>, <header>, <footer>)

Semantic elements improve webpage structure and SEO.

Common Semantic Tags

Tag	Description
<header>	Defines the header section
<nav>	Defines navigation links
<section>	Groups content together
<article>	Represents an independent article
<aside>	Sidebar content
<footer>	Footer section

Example: Webpage Layout Using Semantic Tags

```
<header>
  <h1>My Website</h1>
</header>

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

```
    <a href="about.html">About</a> |
    <a href="contact.html">Contact</a>
</nav>

<section>
    <h2>Welcome</h2>
    <p>This is my website where I share content.</p>
</section>

<article>
    <h3>Latest Blog Post</h3>
    <p>This is an article about web development.</p>
</article>

<footer>
    <p>&copy; 2025 My Website</p>
</footer>
```

---

6. Assignment: Creating a Basic Contact Form and a Simple Webpage Layout

Task 1: Create a Basic Contact Form

1. Open your **code editor**.
2. Create a new file and save it as contact.html.
3. Write the following code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Contact Us</title>
</head>
<body>
  <h1>Contact Us</h1>
  <form>
    <label for="name">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>

    <label for="message">Message:</label><br>
```

```
    <textarea id="message" name="message" rows="5"
cols="40"></textarea><br><br>


    <button type="submit">Send</button>
  </form>
</body>
</html>
```

Task 2: Create a Simple Webpage Layout

1. Open **VS Code** or any editor.
2. Create a file named index.html.
3. Write the following code:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Webpage</title>
</head>
<body>
  <header>
    <h1>Welcome to My Website</h1>
  </header>
```

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

  <section>
    <h2>Introduction</h2>
    <p>This is my simple webpage layout.</p>
  </section>

  <article>
    <h3>My Blog</h3>
    <p>This is a blog post about HTML.</p>
  </article>

  <footer>
    <p>&copy; 2025 My Website</p>
  </footer>
</body>
```

</html>

Expected Output

- A webpage with **a header, navigation menu, content section, and footer**.
- A **contact form** that allows users to enter their name, email, and message.

✅ **Submit your completed HTML files (contact.html and index.html) to the instructor.**

CONCLUSION

In this module, you have learned: ✔ How to add **images** and **links**
✔ How to create **tables** and **forms**
✔ How to use **semantic HTML elements**

🎯 **Next Week:** We will explore **CSS for styling HTML pages.** 🚀

Happy Coding! 🎉

STYLING & ADVANCED HTML CONCEPTS

1. INTRODUCTION TO CSS (BASIC STYLING WITH INLINE AND INTERNAL CSS)

What is CSS?

CSS (Cascading Style Sheets) is used to style HTML elements. It controls the layout, colors, fonts, and overall appearance of a webpage.

Ways to Apply CSS

1. **Inline CSS** – Applied directly inside an HTML element using the style attribute.
2. **Internal CSS** – Defined within the <style> tag inside the <head> section of the HTML document.
3. **External CSS** – Written in a separate .css file and linked to the HTML document.

Example: Inline & Internal CSS

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      background-color: lightblue;
      font-family: Arial, sans-serif;
    }
```

```
    h1 {
      color: darkblue;
      text-align: center;
    }
  </style>
</head>
<body>
  <h1 style="font-size: 30px;">Welcome to My Website</h1>
  <p style="color: green;">This is a paragraph with inline styling.</p>
</body>
</html>
```

✅ **Inline CSS** is useful for quick styling but should be avoided for larger projects.
✅ **Internal CSS** is useful for single-page designs.

---

2. Using and for Page Structure

What are and ?

- ** (Division Tag)** – Used to create containers or sections in a webpage.
- **** – Used to style inline text elements.

**Example: Using and **

```
<!DOCTYPE html>
```

```
<html>
<head>
  <style>
    .container {
      width: 50%;
      background-color: #f4f4f4;
      padding: 10px;
      border: 1px solid #ddd;
    }
    .highlight {
      color: red;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <div class="container">
    <h2>Welcome to My Webpage</h2>
    <p>This is a paragraph inside a `&lt;div&gt;`.</p>
    <p>This is a <span class="highlight">highlighted text</span> inside a paragraph.</p>
```

```
  </div>
</body>
</html>
```

✅ ** ** is a block-level element, often used for layout.
✅ **** is an inline element, used to style specific words.

---

3. Creating Navigation Bars and Footers

Navigation Bar with CSS

```
<!DOCTYPE html>
<html>
<head>
  <style>
    nav {
      background-color: black;
      padding: 10px;
    }
    nav a {
      color: white;
      text-decoration: none;
      margin: 10px;
      padding: 8px 15px;
```

```
      display: inline-block;
    }
    nav a:hover {
      background-color: gray;
    }
  </style>
</head>
<body>
  <nav>
    <a href="#">Home</a>
    <a href="#">About</a>
    <a href="#">Contact</a>
  </nav>
</body>
</html>
```

Footer Example

```
<footer style="background-color: black; color: white; text-align: center; padding: 10px;">
  © 2025 MyWebsite. All Rights Reserved.
</footer>
```

✅ **Navigation bars** provide easy page navigation.
✅ **Footers** contain copyright and contact information.

4. Introduction to Multimedia Elements (<audio>, <video>)

HTML allows you to add audio and video directly into your webpage.

Adding Audio

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

Adding Video

```
<video width="400" controls>
  <source src="video.mp4" type="video/mp4">
  Your browser does not support the video tag.
</video>
```

✅ **controls attribute** adds play, pause, and volume controls.
✅ **Ensure correct file format** for browser compatibility.

---

5. Understanding Meta Tags and SEO Basics

What are Meta Tags?

Meta tags provide **information about a webpage** to search engines and browsers.

Common Meta Tags

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Learn HTML and CSS for web development.">
  <meta name="keywords" content="HTML, CSS, Web Development">
  <meta name="author" content="Your Name">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>SEO Basics</title>
</head>
```

✅ **meta name="description"** – Provides a summary of the webpage.
✅ **meta name="keywords"** – Helps search engines understand your content.
✅ **meta name="viewport"** – Ensures mobile responsiveness.

---

6. Forms with Validation

HTML Form Example

```
<form action="submit.php" method="POST">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required><br><br>
```

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

<label for="password">Password:</label>
<input type="password" id="password" name="password" minlength="6" required><br><br>

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

Form Validation Rules

✅ **required** – Ensures the field cannot be left empty.
✅ **type="email"** – Ensures only valid emails are entered.
✅ **minlength="6"** – Ensures a minimum length for passwords.

---

7. Assignment: Designing a Simple Portfolio Webpage

Task 1: Create a Portfolio Page

1. Create an HTML file **portfolio.html**.
2. Add:
    - A **navigation bar** with "Home," "About," and "Contact."
    - A **hero section** with your name and a brief intro.
    - A **portfolio section** listing your skills and projects.
    - A **contact form** for inquiries.

Example Portfolio Layout

```
<!DOCTYPE html>
<html>
<head>
  <title>My Portfolio</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 0; padding: 0; }
    nav { background: black; padding: 10px; }
    nav a { color: white; text-decoration: none; margin: 10px;
padding: 8px; display: inline-block; }
    .hero { text-align: center; padding: 50px; background: lightgray; }
    .portfolio { padding: 20px; }
    .contact { padding: 20px; background: #f4f4f4; }
  </style>
</head>
<body>

  <nav>
    <a href="#">Home</a>
    <a href="#">About</a>
    <a href="#">Contact</a>
```

```
</nav>

<div class="hero">
  <h1>Welcome to My Portfolio</h1>
  <p>Web Developer | Designer | Freelancer</p>
</div>

<div class="portfolio">
  <h2>My Projects</h2>
  <ul>
    <li>Project 1: Personal Blog</li>
    <li>Project 2: Business Website</li>
  </ul>
</div>

<div class="contact">
  <h2>Contact Me</h2>
  <form>
    <label for="name">Name:</label>
    <input type="text" id="name" required><br><br>
```

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

        <input type="submit" value="Send Message">
    </form>
  </div>

</body>
</html>
```

✅ **Submit your portfolio webpage for review.**

🚀 **Next Lesson:** Responsive & Interactive HTML Elements.

RESPONSIVE & INTERACTIVE HTML

1. INTRODUCTION TO RESPONSIVE WEB DESIGN

What is Responsive Web Design?

Responsive Web Design (RWD) ensures that a website adapts to different screen sizes and devices (desktops, tablets, and smartphones). It is achieved using **CSS media queries, flexible grids, and responsive images**.

Key Techniques for Responsive Design

1. **Using the viewport meta tag**

<meta name="viewport" content="width=device-width, initial-scale=1.0">

   - Ensures proper scaling on mobile devices.

2. **Using percentage-based widths instead of fixed widths**

```
.container {
  width: 80%;  /* Instead of using pixels */
  margin: auto;
}
```

Using media queries

```
@media screen and (max-width: 768px) {
  body {
    background-color: lightgray;
  }
}
```

- Changes the background color when screen width is **less than 768px**.

3. **Using Flexbox and Grid for Layouts**
   - **Flexbox** Example:

```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

CSS Grid Example:

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

✅ **Why Responsive Design?**

- Provides a **better user experience**
- Improves **SEO ranking** (Google prefers mobile-friendly sites)
- Makes websites **accessible on all devices**

---

2. Embedding Google Maps, YouTube Videos, and Social Media Links

Embedding Google Maps

<iframe

```
  src="https://www.google.com/maps/embed?...YOUR-LOCATION..."
  width="600" height="450" style="border:0;" allowfullscreen>
</iframe>
```

Embedding YouTube Videos

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

Adding Social Media Links

```
<a href="https://www.facebook.com/yourpage" target="_blank">Facebook</a>
<a href="https://twitter.com/yourprofile" target="_blank">Twitter</a>
```

✅ **Using icons for social links**

```
<a href="https://www.instagram.com/yourprofile" target="_blank">
  <img src="instagram-icon.png" width="30px">
</a>
```

---

3. BASICS OF HTML5 (<CANVAS>, <SVG>)

Using <canvas> for Graphics

- <canvas> is used to draw graphics, animations, and charts dynamically using JavaScript.

```
<canvas id="myCanvas" width="400" height="200" style="border:1px solid black;"></canvas>


<script>
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
  ctx.fillStyle = "blue";
  ctx.fillRect(50, 50, 100, 100);
</script>
```

Using <svg> for Scalable Vector Graphics

- <svg> is used to create vector graphics that do not lose quality when resized.

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
</svg>
```

✅ **Why use Canvas & SVG?**

- **Canvas** is best for real-time graphics like games.
- **SVG** is best for icons and logos since it scales without losing quality.

---

4. Introduction to Bootstrap for Quick Styling

What is Bootstrap?

Bootstrap is a CSS framework that makes designing responsive websites **easier and faster**. It provides **ready-made grids, buttons, navigation bars, and more**.

How to Add Bootstrap to Your HTML Page?

Add this line in the <head> section:

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
```

Basic Bootstrap Layout

```
<div class="container">
  <h1 class="text-center text-primary">Welcome to Bootstrap</h1>
  <p class="lead">This is a simple example using Bootstrap classes.</p>
  <button class="btn btn-success">Click Me</button>
</div>
```

Creating a Responsive Grid with Bootstrap

```
<div class="container">
  <div class="row">
    <div class="col-md-6">Left Column</div>
    <div class="col-md-6">Right Column</div>
  </div>
</div>
```

✅ Why Bootstrap?

- Saves time with **pre-built components**

- Mobile-friendly **by default**
- Provides a **consistent layout across all browsers**

---

5. Hosting a Basic HTML Page on GitHub Pages

What is GitHub Pages?

GitHub Pages allows you to host static websites **for free** using your GitHub repository.

Steps to Host a Webpage on GitHub

1. **Create a GitHub Repository**
   - Go to GitHub and create a new repository.
2. **Upload Your HTML Files**
   - Click on "Upload files" and select your HTML, CSS, and JavaScript files.
3. **Enable GitHub Pages**
   - Go to **Settings** → **Pages**
   - Under **Branch**, select main and click **Save**.
4. **Access Your Website**
   - Your site will be available at: https://your-username.github.io/your-repository-name/

✅ **GitHub Pages is great for:**

- Hosting **personal portfolios**
- Sharing **projects and documentation**
- Deploying **simple static websites**

6. Final Project: Creating a Complete Multi-Page Website

Project Requirements

- **Homepage:** A welcome page with an introduction.
- **About Page:** Information about yourself or the company.
- **Portfolio Page:** A gallery showcasing projects.
- **Contact Page:** A form for user inquiries.

Example Multi-Page Website Structure

```
/my-website
    ├── index.html
    ├── about.html
    ├── portfolio.html
    ├── contact.html
    ├── styles.css
    ├── images/
    └── scripts.js
```

Homepage Example (index.html)

```
<!DOCTYPE html>
<html>
<head>
  <title>My Website</title>
  <link rel="stylesheet" href="styles.css">
```

```
</head>
<body>
  <header>
    <h1>Welcome to My Website</h1>
  </header>
  <nav>
    <a href="index.html">Home</a>
    <a href="about.html">About</a>
    <a href="portfolio.html">Portfolio</a>
    <a href="contact.html">Contact</a>
  </nav>
  <section>
    <p>This is the homepage of my first multi-page website.</p>
  </section>
</body>
</html>
```

Project Submission Guidelines

1. Upload your project to **GitHub** or share a ZIP file.
2. Make sure all pages are linked correctly.
3. Use **responsive design** principles.
4. Include **Bootstrap for styling**.

CONCLUSION

✅ You have now learned:

- Responsive Web Design
- Embedding Multimedia and Social Media Links
- Using HTML5 <canvas> and <svg>
- Quick Styling with Bootstrap
- Hosting Websites on GitHub Pages
- Building a Complete Multi-Page Website

🎯 **Next Step:** Start your journey into **JavaScript for interactivity**! 🚀