Week One

Date: 22-03-2025

Module 1

Introduction to Web Development & HTML Basic

What is HTML?

HTML (Hyper Text Markup Language) is the most basic building block of the web. It define teh meaning and structure of web content other technologies beside HTML are generally used to describe a web page's appearance/presentation (CSS) or functionally/behaviour (JavaScript)

STRUCTURE OF AN HTML DOCUMENT

- <!DOCTYPE html>
- <html>
- <head>
- <body>

```
<!DOCTYPE html>

<html>

        <head>

                <title> Title Here</title>

        </head>

        <body>

                Web Page Content Goes Here.

        </body>

</html>
```

Explanation

<!DOCTYPE html>: Declares the document type.

<html>: The root element.

<head>: Contains meta-information (title, links, scripts).

<title>: Sets the title of the webpage.

<body>: Contains the visible content.

Basic HTML Elements

Headings, Paragraphs, and Text Formatting:

```
<h1>Heading 1</h1>

<h2>Heading 2</h2>

<p>This is a paragraph. <strong>Bold text</strong> and <em>italic text</em>. </p>
```

<h1> to <h6>: Headings (largest to smallest).

<p>: Paragraphs.

<strong>: Bold text.

<em>: Italic text.

Head Tags
<title>
<meta>
<script>
<style>

Content Tags
<h1> to <h6>
<span>
<strong>
<div>

List Tags
<ul>
<ol>
<li >

Important Tags

Empty Tags
<br>
<hr>
<img>
<input>
<link>
<meta>

Lists

- **Ordered List**
- **Unordered List**

Ordered List

```
<ol>
  <li>First item</li>
  <li>Second item</li>
</ol>
```

Unordered List

```
<ul>
  <li>First item</li>
  <li>Second item</li>
</ul>
```

Explanation

<!DOCTYPE html>: Declares the document type.

<html>: The root element.

<head>: Contains meta-information (title, links, scripts).

<title>: Sets the title of the webpage.

<body>: Contains the visible content.

Links, Images, and Buttons:

Link

- `<a href="https://www.example.com">Visit Example</a>`
- `<a href="https://www.example.com"  target="_blank">Open in new tab</a>`
- `<a href="mailto:example@example.com">Send  Email</a>`
- `<a href="#section1">Jump to Section 1</a>`

  href=>Hypertext Reference

Images

```
<img src="image.jpg" alt="Description og image">

<img src="image.jpg" alt="Description og image" height="300" width="300">
```

Src => External/internal resource / source url

Button

```
<button>Submit</button>
```

Block and Inline Elements

- Block Elements

```
<div>This is a block element. </div>

<article>This is a block element. </article>

<section>This is a block element. </section>

<header>This is a block element. </header>

<footer>This is a block element. </footer>
```

- Inline Elements

```
<span>This is a inline element. </span>

<a href="#">This is a inline element. </a>

<strong>This is a inline element. </strong>

<em>This is a inline element. </em>
```

Week Two

Date: 25-03-2025 To 29-03-2025

Module 2

HTML Forms, Tables & Semantic Elements

HTML Forms

1. Introduction to Forms

- Forms are used to collect user input and send it to a server for processing.
- Basic Structure:

```
<form action="submit.php" method="POST">
  <!-- Input fields go here -->
</form>
```

2. Input Fields and Labels

- Text Fields:

```
<label for="name">Name:</label>
<input type="text" id="name" name="name" placeholder="Enter your name">
```

- Password Field:

```
<label for="name">Name:</label>
<input type="text" id="name" name="name" placeholder="Enter your name">
```

3. Checkboxes and Radio Buttons

- Checkbox (Multiple Selections):

```
<label>
 <input type="checkbox" name="subscribe" value="yes"> Subscribe to newsletter
</label>
```

- Radio Buttons (Single Selection):

```
<p>Gender:</p>
<label>
  <input type="radio" name="gender" value="male"> Male
</label>
<label>
  <input type="radio" name="gender" value="female"> Female
</label>
```

4. Dropdown

- Create a dropdown menu:

```
<label for="country">Country:</label>
<select id="country" name="country">
  <option value="usa">USA</option>
  <option value="canada">Canada</option>
  <option value="uk">UK</option>
```

5. Submit Button and Form Submission

- **Submit Button:** Sends form data to the server.

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

- **Action Attribute:** Specifies where the form data is sent.

```
<form action="submit.php" method="POST">
```

METHOD ATTRIBUTE:

- **GET:** SENDS DATA IN THE URL (LESS SECURE, GOOD FOR SEARCH QUERIES).
- **POST:** SENDS DATA IN THE BODY (MORE SECURE, USED FOR SENSITIVE INFO).

HTML Chapter: Video, Audio, Embed, and Tables

Video in HTML

HTML allows embedding videos using the `<video>` tag. The `<video>` element supports multiple formats such as MP4, WebM, and Ogg.

1.1 Syntax

```
<video width="600" controls>

 <source src="video.mp4" type="video/mp4">

 <source src="video.webm" type="video/webm">

 Your browser does not support the video tag.

</video>
```

1.2 Attributes

- controls: Displays video controls (play, pause, volume, etc.).
- autoplay: Starts playing automatically when the page loads.
- loop: Repeats the video continuously.
- muted: Mutes the video by default.
- poster: Specifies an image to display before the video starts playing.

1.3 Exercises

1. Create a simple HTML page that embeds a video using the `<video>` tag.
2. Add multiple source files to ensure browser compatibility.
3. Use the `autoplay` and `muted` attributes and observe their behavior.
4. Apply the `poster` attribute with an image before the video plays.
5. Embed a video and style it with CSS for responsiveness.

---

2. Audio in HTML

HTML allows embedding audio using the `<audio>` tag, similar to the `<video>` tag.

2.1 Syntax

```
<audio controls>

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

 <source src="audio.ogg" type="audio/ogg">

 Your browser does not support the audio element.

</audio>
```

2.2 Attributes

1. controls: Adds play, pause, and volume controls.
2. autoplay: Plays audio automatically.
3. loop: Loops the audio after it ends.
4. muted: Mutes the audio by default.

2.3 Exercises

1. Embed an audio file using the `<audio>` tag.
2. Add multiple source formats for compatibility.
3. Use the `loop` attribute and test the looping behavior.
4. Create an audio player with CSS styling.
5. Combine audio and video elements on a single webpage.

---

3. Embed in HTML

The `<embed>` tag is used to integrate external content such as PDFs, videos, and applications.

3.1 Syntax

```
<embed src="document.pdf" width="600" height="400" type="application/pdf">
```

3.2 Common Uses

- Embedding YouTube videos

- Displaying PDFs

- Integrating external applications

3.3 Exercises

1. Embed a PDF file using the `<embed>` tag.
2. Embed a YouTube video using an `<iframe>`.
3. Embed an external interactive map.
4. Experiment with different `width` and `height` attributes.
5. Create a webpage with both embedded video and a document.

---

4. HTML Tables

Tables are used to organize data into rows and columns.

4.1 Basic Table Structure

```
<table border="1">
 <tr>
  <th>Name</th>
  <th>Age</th>
  <th>City</th>
 </tr>
 <tr>
  <td>Alice</td>
  <td>25</td>
  <td>New York</td>
 </tr>
</table>
```

4.2 Adding Borders, Padding, and Spacing

```
<style>
 table {
  border-collapse: collapse;
  width: 100%;
 }
 th, td {
  border: 1px solid black;
  padding: 10px;
  text-align: left;
 }
</style>
```

4.3 Merging Cells with `rowspan` and `colspan`

```
<table border="1">

 <tr>

  <th colspan="2">Full Name</th>

  <th>Age</th>

 </tr>

 <tr>

  <td>John</td>

  <td>Doe</td>

  <td>30</td>

 </tr>

</table>
```

4.4 Exercises

1. Create a basic table with 3 columns and 3 rows.
2. Add borders, padding, and spacing to a table.
3. Use `colspan` to merge header cells.
4. Use `rowspan` to merge row cells.
5. Design a responsive table using CSS.

HTML5 Semantic Elements

Understanding HTML5 Semantic Elements

Semantic elements in HTML5 improve the readability, accessibility, and SEO of web pages. They provide meaningful structure to a webpage.

Common Semantic Elements

1. <HEADER>

- Defines introductory content or navigation links at the top of a page or section.
- Usually contains the logo, navigation menu, and site title.

2. <NAV>

- Represents a block of navigation links.
- Typically contains menus, links to internal pages, or external sites.

3. <SECTION>

- Represents a thematic grouping of content.
- Can be used to divide content into logical sections.

4. <ARTICLE>

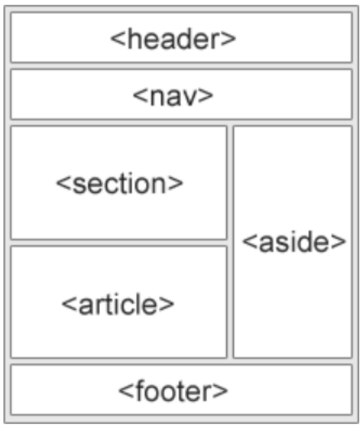
- Contains independent, self-contained content such as blog posts, news articles, or forum posts.

5. <ASIDE>

- Represents content related to the main content, like sidebars or advertisements.

6. <FOOTER>

- Defines the footer of a webpage or section.
- Usually contains copyright information, contact details, or social media links.



Benefits of Semantic HTML

1. Improved SEO – Search engines understand the content structure better.

2. Better Accessibility – Screen readers can interpret content more effectively.

3. Easier Maintenance – Code is more readable and maintainable.

4. Consistent Styling – Enables better styling with CSS.

Exercises for HTML5 Semantic Elements

1. Create an HTML page with `<header>`, `<nav>`, `<section>`, `<article>`, `<aside>`, and `<footer>`.

2. Modify an existing webpage by replacing `<div>` elements with semantic elements.

3. Create a blog page using `<article>` for posts and `<aside>` for additional content.

4. Add a `<nav>` section with a horizontal menu using `<ul>` and `<li>`.

5. Design a webpage layout using only semantic elements and validate it using the W3C validator.

Week Three

Date: 04-04-2025

Module 3

Introduction to CSS & Styling Basics

CSS Basics

Types of CSS

1. **Inline CSS**
   - Applied directly within the HTML element using the `style` attribute.

Example:

```
<p style="color: blue; font-size: 16px;">This is an inline CSS example.</p>
```

Best for: Quick fixes but should be avoided for large-scale styling.

2. **Internal CSS**
   - Defined within a `<style>` block in the `<head>` section.

Example:

```
<style>
  p { color: green; font-size: 18px; }
</style>
```

Best for: Single-page styling.

3. **External CSS**

- Linked through an external file using `<link>`.

```
Example:

<link rel="stylesheet" href="styles.css">
```

Best for: Large projects and consistent styling across multiple pages.

CSS Selectors

1. **Element Selector**
   - Targets all elements of a specific type.

```
Example:

p { color: red; }
```

2. **Class Selector (`.`)**
   - Targets elements with a specific class.

```
Example:

.highlight { background-color: yellow; }
```

3. **. ID Selector (`#`)**
   - Targets a single element with a specific ID.

```
Example:

#unique { font-size: 20px; }
```

4. **. Universal Selector (`*`)**
    - Targets a single element with a specific ID.

Example:

```
* {
  border: 2px solid green;
  background-color: beige;
}
```

Styling Text

- Fonts: `font-family: Arial, sans-serif;`
- Colors: `color: blue;`
- Text Alignment: `text-align: center;`

CSS Box Model

- The CSS box model defines the spacing around elements.

1. **Margin – Space outside the element.**
2. **Border – Defines the boundary of an element.**
3. **Padding – Space inside the element, between content and border.**
4. **Content – The actual content inside the box.**

Example:

```
.box {
  margin: 20px;
  border: 2px solid black;
  padding: 15px;
  width: 200px;
}
```

Exercises for CSS Basics

1. Create a webpage with inline, internal, and external CSS applied to different elements.
2. Use class and ID selectors to style a webpage differently.
3. Style a paragraph with different font styles, colors, and text alignment.
4. Apply the box model to a `div` and experiment with margins, padding, and borders.
5. Create a navigation menu using CSS for styling.

Practical Implementation

Final Project: Building a Simple Web Page

Apply all the learned concepts to create a structured webpage:

- Use semantic HTML to organize the content.

- Apply CSS styling using external stylesheets.

- Implement navigation, sections, and articles**.

- Style the page using CSS selectors and the **box model**.

Final Exercise

1. Design a personal blog page with semantic HTML and CSS.

2. Implement a navigation bar styled using CSS.

3. Style different sections with unique colors, fonts, and layouts.

4. Ensure the page is responsive and visually appealing.

5. Validate the HTML and CSS using online validation tools.

CSS Layout Techniques

1. Flexbox

- A one-dimensional layout model used for distributing space between items in a container.

- **Properties:**

  1. display: flex;
  2. justify-content (aligns items horizontally)
  3. align-items (aligns items vertically)

2. CSS Grid

- A two-dimensional layout system that allows arranging elements into rows and columns.
- **Properties:**

  i. display: grid;
  ii. grid-template-columns & grid-template-rows
  iii. gap (controls spacing)

3. Floats

- Elements float to the left or right, with text wrapping around them.
- Commonly used with float: left; or float: right;.
- Requires clear: both; to prevent layout issues.

4. Positioning & Display

- display: block;, display: inline;, display: flex;, display: grid;
- position: static;, position: relative;, position: absolute;, position: fixed;

Exercises

1. Create a webpage layout using Flexbox to arrange three sections: header, main content, and footer.
2. Implement a CSS Grid layout with 3 columns and 2 rows.
3. Use float to position an image to the left with text wrapping around it.
4. Experiment with display: inline-block; and display: grid; to create navigation menus.
5. Adjust justify-content and align-items properties in Flexbox for a centered layout.

Block vs. Inline Elements

- Elements in HTML are categorized into **block-level** and **inline-level** elements.

Block Elements

- Always start on a new line and take up the full width available.
- Common block elements: <div>, <p>, <h1> - <h6>, <section>, <article>

Inline Elements

- Do not start on a new line and only take up as much width as needed.
- Common inline elements: <span>, <a>, <strong>, <em>

Exercises

1. Convert a block-level <div> into an inline element using display: inline;.
2. Create a navigation bar using inline elements.
3. Apply a width to an inline element and observe the effect.
4. Nest inline elements inside a block element and apply styles.
5. Change an inline element to display: block; and observe the behavior.

Width, Height, and Overflow Properties

- These properties control the dimensions and behavior of elements.

Width & Height

- width: auto; (default behavior)
- height: auto; (default behavior)
- max-width & max-height restrict size without overflow.
- min-width & min-height ensure a minimum dimension.

Overflow Property

- Controls how content behaves when it exceeds its container’s size.
- overflow: visible; (default, content spills out)
- overflow: hidden; (hides overflowed content)
- overflow: scroll; (adds scrollbars)
- overflow: auto; (adds scrollbars only if necessary)

Exercises

1. Set a fixed width and height for a <div> and experiment with different overflow properties.
2. Use max-width and min-width to create a responsive image gallery.
3. Create a scrolling container using overflow: scroll;.
4. Apply height: 100vh; to an element and observe the effect.
5. Use min-height on a section and adjust content dynamically.

Positioning Elements: Static, Relative, Absolute, Fixed

- Positioning determines how an element is placed within a document.

Static Positioning (default)

- Elements follow the normal document flow.
- Example: position: static;

Relative Positioning

- Elements are positioned relative to their normal position.
- Example: position: relative; top: 20px; left: 30px;

Absolute Positioning

- Elements are removed from normal flow and positioned relative to their nearest positioned ancestor.
- Example: position: absolute; top: 50px; left: 100px;

Fixed Positioning

- Elements remain fixed at a specific location on the screen.
- Example: position: fixed; bottom: 0; right: 0;

Exercises

1. Position a <div> absolutely within a relative parent container.
2. Create a sticky header using position: fixed;.
3. Experiment with position: relative; and move an element 50px right and 30px down.
4. Use position: absolute; to place an icon inside a button.
5. Create a sidebar that remains in place using position: fixed;.