Class 1: Introduction to HTML

SESSION OVERVIEW

By the end of this session, students will be able to:

- Understand what HTML is and its importance in web development.
- Identify and analyze the structure of an HTML document and its core elements.
- Use basic HTML tags such as headings, paragraphs, lists, and links to create structured content.
- Apply attributes to HTML elements to modify their behavior and appearance.
- Develop a fundamental understanding of HTML as the foundation for building web pages.

Fundamental Web Concepts

1. Webpages and Websites

What is a Webpage?

- A **webpage** is a single document on the internet that can contain **text, images, videos, and interactive elements**.
- Webpages are created using **HTML (HyperText Markup Language)** and are styled with **CSS (Cascading Style Sheets)**.
- Webpages can be either **static** (fixed content) or **dynamic** (interactive, updating in real-time).

What is a Website?

- A **website** is a **collection of interconnected webpages** grouped under a **domain name** and hosted on a server.
- Websites can be **personal, corporate, e-commerce, or web applications**.

Difference Between Webpage and Website

Feature	Webpage	Website
Definition	A single document	A collection of multiple webpages
Example	A blog post	A complete blog with multiple posts
URL	`example.com/about.html`	`example.com`

---

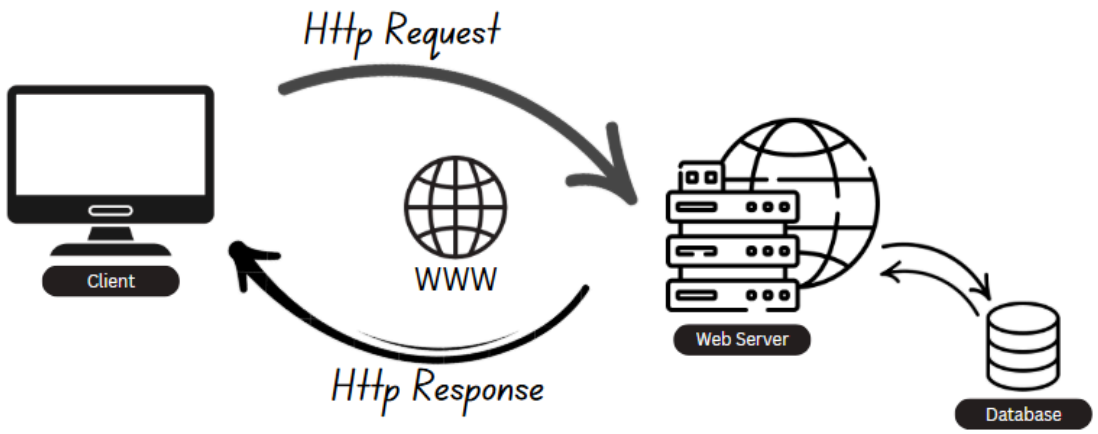
2. Client-Server Architecture

What is Client-Server Architecture?

- The **client-server model** is a computing structure where the **client (browser or app)** requests data, and the **server** responds with the requested information.

How It Works?

1. **Client sends a request** (e.g., loading a webpage).
2. **Server processes the request** and retrieves data if needed.
3. **Database stores website data**.
4. **Server sends a response** with the requested page or data.
5. **Client displays the webpage** to the user.



Above, you can see a **diagram explaining how the client-server architecture works.**

Example:

- When you **visit Google**, your **browser (client)** sends a request to **Google's servers**.
- Google's server **processes the request** and sends the **Google homepage** back to your browser.

---

3. Web Browsers

What is a Web Browser?

- A **web browser** is a software application that allows users to **access websites** on the **World Wide Web (WWW)**.
- Popular browsers include **Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, and Opera**.

Functions of a Web Browser

- **Rendering Webpages:** Parses HTML, CSS, and JavaScript to display content.
- **Caching Data:** Stores web resources for faster future access.
- **Managing Secure Connections:** Uses **HTTPS and SSL/TLS** for security.
- **Interpreting JavaScript:** Executes scripts to enable interactivity.

How a Web Browser Works?

1. The user **enters a URL** in the browser's address bar.
2. The browser **sends an HTTP request** to the website's server.
3. The **server sends back an HTML document** with additional resources (CSS, JavaScript).
4. The **browser processes and renders the page** for the user.

---

4. World Wide Web (WWW)

What is the WWW?

- The **World Wide Web (WWW)** is a **system of interlinked web documents** that can be accessed via the **Internet** using web browsers.
- Invented by **Tim Berners-Lee in 1989**.

How the WWW Works?

1. Websites are stored on **web servers** worldwide.
2. Users access webpages through **URLs** in a browser.
3. The browser fetches the requested pages using **HTTP/HTTPS protocols**.

WWW vs. Internet

- **Internet:** The **physical network** that connects computers worldwide.
- **WWW:** The **system of web pages** that exist on the internet.

---

5. URLs (Uniform Resource Locator)

What is a URL?

- A **URL (Uniform Resource Locator)** is the **address of a webpage** or **resource** on the internet.

Parts of a URL

Example: `https://www.example.com/page.html`

- **Protocol:** `https://` (Defines communication rules like HTTP or HTTPS).
- **Domain Name:** `www.example.com` (The website's unique name).
- **Path:** `/page.html` (The specific resource being requested).

HTTP vs. HTTPS

- **HTTP (Hypertext Transfer Protocol):** Standard protocol for website communication.
- **HTTPS (Secure HTTP):** Uses **SSL/TLS encryption** for data security.

---

6. Domain Name System (DNS)

What is DNS?

- The **Domain Name System (DNS)** translates human-readable domain names into **IP addresses** used by computers.

How DNS Works?

1. The user enters a **website URL** (e.g., `www.google.com`).
2. The browser queries a **DNS server** to get the **IP address**.
3. The browser connects to the **server using the IP address** and loads the website.

Why is DNS Important?

- Allows users to **access websites using easy-to-remember names** instead of complex IP addresses.
- Improves **website loading speed** through DNS caching.

Types of DNS Records

- **A Record:** Maps a domain name to an IP address.
- **CNAME Record:** Redirects one domain to another.
- **MX Record:** Used for email routing.

codingninjas

Fundamentals of HTML

1. Setting Up VS Code

Why Use VS Code?

VS Code (Visual Studio Code) is a powerful **code editor** with features like **syntax highlighting, IntelliSense, live preview, and extensions** for easier web development.

Steps to Set Up VS Code for HTML Development:

1. **Download and Install VS Code** from the official website: https://code.visualstudio.com/
2. **Install Extensions:**
    - **Live Server** (to preview HTML files in the browser).
    - **Prettier** (for code formatting).
    - **HTML Boilerplate** (to generate a basic HTML structure).
3. **Create an HTML File:**
    - Open VS Code → Click on **File** → **New File** → Save it as `index.html`.
4. **Run the HTML File Using Live Server:**
    - Right-click the file and select **"Open with Live Server"** to preview it in a browser.

---

2. HTML Structure

What is HTML?

HTML (**HyperText Markup Language**) is the standard language used to create webpages. It consists of elements enclosed in **tags (`<tagname>`)** that define the structure and content of a page.

Basic HTML Structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="A sample webpage for learning HTML.">
    <title>My First HTML Page</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is a simple HTML document.</p>
```

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

Explanation of Key HTML Elements:

- **`<!DOCTYPE html>`** – Declares the document type (HTML5).
- **`<html>`** – The root element of the webpage.
- **`<head>`** – Contains metadata and links to styles/scripts.
- **`<meta>`** – Provides additional information (character encoding, description, responsiveness).
- **`<title>`** – Defines the title shown on the browser tab.
- **`<body>`** – Contains the visible content of the webpage.

---

3. Importance of Head, Meta, and Body Tags in SEO Optimization

1. <head> Section (Essential for SEO and Page Setup)

- Contains metadata, links to stylesheets, and external resources.
- Helps search engines understand page content.

Example of SEO-friendly <head> Section:

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Learn the fundamentals of HTML with
examples and best practices.">
    <meta name="keywords" content="HTML, web development, programming">
    <meta name="author" content="John Doe">
    <title>HTML Fundamentals Guide</title>
</head>
```

2. <meta> Tags (Improving Search Rankings)

- **`<meta name="description">`** – Helps search engines display a page summary in search results.

- **`<meta name="keywords">`** – Specifies relevant search keywords (not as relevant today).
- **`<meta name="author">`** – Specifies the author of the webpage.
- **`<meta name="viewport">`** – Ensures the webpage is responsive on mobile devices.

Question: How Does Metadata Help in SEO?

Metadata provides **important information** about a webpage to **search engines** and **users**, improving search rankings and click-through rates (CTR).

Key Meta Tags for SEO:

- **Title Tag (`<title>`)** – Defines the page title and helps in ranking.
- **Meta Description (`<meta name="description">`)** – Summarizes page content and improves CTR.
- **Meta Robots (`<meta name="robots">`)** – Controls whether search engines index or follow links.
- **Meta Viewport (`<meta name="viewport">`)** – Ensures mobile-friendliness, improving mobile SEO.

3. <body> Section

- The main part of the HTML document that contains the content users see.
- Includes **text, images, links, tables, forms, multimedia, and interactive elements**.

4. Introduction to Essential HTML Tags

1. What are HTML Tags?

HTML (**HyperText Markup Language**) uses **tags** to define elements in a webpage. Tags are enclosed in **angle brackets (<>)** and help browsers understand the **structure and content** of the webpage.

Types of HTML Tags

1. **Opening and Closing Tags:**
    - Most HTML elements come in **pairs**: an **opening tag (`<tag>`)** and a **closing tag (`</tag>`)**.
    - The closing tag has a **forward slash (`/`)** before the tag name.

Example:

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

2. **Self-Closing Tags (Void Elements):**
    - Some HTML elements **do not require** a closing tag.
    - They are called **self-closing tags** or **void elements**.

Example:

```
<br>  <!-- Line break -->

<hr>  <!-- Horizontal line -->

<img src="image.jpg" alt="Example Image">
```

2. Basic HTML Tags and Their Usage

Tag	Type	Purpose	Example
`<h1>` to `<h6>`	Opening & Closing	Defines headings (largest to smallest)	`<h1>Main Heading</h1>`
`<p>`	Opening & Closing	Defines a paragraph	`<p>This is a paragraph.</p>`
`<b>` / `<strong>`	Opening & Closing	Bold text (important for readability)	`<strong>Important!</strong>`
`<i>` / `<em>`	Opening & Closing	Italicized/emphasized text	`<em>Emphasized Text</em>`
`<u>`	Opening & Closing	Underlined text	`<u>Underlined</u>`
` `	Self-Closing	Inserts a line break	`First line Second line`
`<hr>`	Self-Closing	Adds a horizontal line	`<hr>`
`<a>`	Opening & Closing	Creates a hyperlink	`<a href="https://example.com">Click Here</a>`
`<div>`	Opening & Closing	Defines a block-level section	`<div>This is a section</div>`
`<span>`	Opening & Closing	Defines an inline container	`<span>Inline text</span>`

---

3. Understanding Opening & Closing Tags in Detail

1. Opening & Closing Tags in Action

```
<h1>Welcome to My Website</h1>
<p>This is a <strong>bold</strong> and <em>italic</em> example.</p>
<a href="https://www.google.com">Visit Google</a>
```

- `<h1> ... </h1>` → A heading tag that requires both an **opening (<h1>)** and a **closing (</h1>)** tag.
- `<p> ... </p>` → A paragraph tag that defines a block of text.
- `<a href="https://www.google.com"> ... </a>` → A hyperlink that **requires an `href` attribute**.

---

2. Self-Closing Tags in Action

```
<p>This is a paragraph.</p>
<br>  <!-- Line Break -->
<hr>  <!-- Horizontal Line -->
<img src="example.jpg" alt="Example Image">
```

- `<br>` → Creates a **line break** and does **not** require a closing tag.
- `<hr>` → Inserts a **horizontal line** between sections.
- `<img>` → Loads an **image** and uses the **`src`** attribute to specify the image file.

5. Lists in HTML

Lists are used to organize and display content in a **structured format**. HTML provides three types of lists:

---

1. Ordered List (`<ol>`) – Numbered List

Definition:

An **ordered list** is a list where items are **automatically numbered**. Each list item is wrapped in `<li>` (list item) tags, and the `<ol>` tag defines the list.

Example:

```
<ol>
    <li>Step 1: Open VS Code</li>
    <li>Step 2: Create an HTML file</li>
    <li>Step 3: Write HTML Code</li>
</ol>
```

Types of Ordered Lists (Using `type` Attribute)

You can modify how the numbers are displayed using the `type` attribute:

- **Numbers (1, 2, 3)** – Default
- **Uppercase letters (A, B, C)** – `type="A"`
- **Lowercase letters (a, b, c)** – `type="a"`
- **Roman numerals (I, II, III)** – `type="I"`
- **Lowercase Roman numerals (i, ii, iii)** – `type="i"`

Example with Different Types:

```
<ol type="A">
    <li>Item A</li>
    <li>Item B</li>
</ol>

<ol type="i">
    <li>Item i</li>
    <li>Item ii</li>
</ol>
```

---

2. Unordered List (`<ul>`) – Bullet Points

Definition:

An **unordered list** is a list where items are displayed **with bullet points** instead of numbers. The `<ul>` tag is used for unordered lists, and `<li>` defines each list item.

Example:

```
<ul>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Oranges</li>
</ul>
```

Types of Bullet Points (Using `type` Attribute)

By default, an unordered list uses **disc bullets (•)**, but you can change it using the `type` attribute:

- **Disc (●) – Default**
- **Circle (○) – `type="circle"`**

- Square (■) – `type="square"`

---

3. Definition List (`<dl>`) – Term and Description

Definition:

A **definition list** is used for defining terms along with their descriptions. It consists of:

- `<dl>` – The container for the list.
- `<dt>` – Defines the term (name of the item).
- `<dd>` – Describes the term (definition).

Example:

```
<dl>
    <dt>HTML</dt>
    <dd>HyperText Markup Language - used to structure web pages.</dd>

    <dt>CSS</dt>
    <dd>Cascading Style Sheets - used for styling web pages.</dd>
</dl>
```

6. Media Elements in HTML: Images, Audio, and Video

Media elements in HTML allow developers to **embed visual and audio content** into webpages, enhancing user experience. These elements include **images (`<img>`), responsive images (`<picture>`), audio (`<audio>`), and video (`<video>`)**.

---

1. Adding an Image (`<img>`)

Definition:

The `<img>` tag is used to **embed images** in an HTML document. It does not have a closing tag since it is a **self-closing element**.

Example:

```
<img src="image.jpg" alt="A beautiful scenery" width="400">
```

Attributes:

- **`src`** – Specifies the image file location.
- **`alt`** – Alternative text for screen readers (important for **accessibility & SEO**).
- **`width` & `height`** – Adjust the image dimensions.

---

2. Using the `<picture>` Element for Responsive Images

Definition:

The `<picture>` element is used to provide **responsive images**, meaning different images can be loaded based on **screen size or device type**.

Example:

```
<picture>
    <source srcset="small.jpg" media="(max-width: 600px)">
    <source srcset="large.jpg" media="(min-width: 601px)">
    <img src="default.jpg" alt="Responsive Image">
</picture>
```

How It Works?

- The **browser selects the most suitable image** based on the device's screen width.
- If the screen width is **600px or less**, `small.jpg` will be displayed.
- If the screen width is **601px or more**, `large.jpg` will be used.
- If none of the conditions match, the browser **falls back** to the `<img>` tag (`default.jpg`).

Why Use `<picture>`?

- Improves **loading speed** by serving smaller images for mobile users.
- Enhances **performance and user experience** by optimizing images for different screen sizes.

---

3. Embedding an Audio File (`<audio>`)

Definition:

The `<audio>` tag allows you to **embed audio files** into a webpage. It provides built-in **playback controls** like play, pause, and volume adjustment.

Example:

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

Attributes:

- **`controls`** – Displays **play/pause buttons** for user interaction.
- **`source`** – Specifies the audio file's location and type.
- **`autoplay`** – Automatically starts playing the audio when the page loads (**not recommended** for user experience).
- **`loop`** – Repeats the audio indefinitely.
- **`muted`** – Starts the audio in a **muted state**.

---

4. Adding a Video File (`<video>`)

Definition:

The `<video>` tag allows **embedding video files** directly into a webpage. It provides user-friendly controls to **play, pause, and adjust volume**.

Example:

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

Attributes:

- **`controls`** – Enables play, pause, and volume controls.
- **`width & height`** – Sets the video dimensions.
- **`autoplay`** – Starts playing automatically (**not recommended**).
- **`loop`** – Repeats the video when it ends.
- **`muted`** – Mutes the video when it loads.
- **`poster`** – Specifies an image to display **before the video starts playing**

7. Tables and Forms in HTML

Tables and forms are essential elements in HTML for **displaying structured data** and **collecting user input**.

1. Creating a Table (`<table>`)

Definition: A table in HTML is created using the `<table>` element, which organizes data into **rows (`<tr>`)** and **columns (`<th>` and `<td>`)**.

Tag	Definition	Example Usage
`<table>`	Creates a table	`<table> ... </table>`
`<tr>`	Defines a row in the table	`<tr> ... </tr>`
`<th>`	Defines a table header (bold & centered)	`<th>Column Name</th>`
`<td>`	Defines a data cell in a row	`<td>Row Data</td>`
`border="1"`	Adds a border to the table (optional)	`<table border="1">`
`colspan`	Merges multiple columns into one	`<td colspan="2">Merged</td>`
`rowspan`	Merges multiple rows into one	`<td rowspan="2">Merged</td>`

Example Table Code:

```
<table border="1">
    <tr>
        <th>Name</th>
        <th>Age</th>
        <th>Country</th>
    </tr>
    <tr>
        <td>Alice</td>
        <td>25</td>
        <td>USA</td>
    </tr>
    <tr>
        <td>Bob</td>
        <td>30</td>
        <td>UK</td>
    </tr>
</table>
```

Example with `colspan` and `rowspan`:

```html
<table border="1">
    <tr>
        <th colspan="2">Full Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>Alice</td>
        <td>Smith</td>
        <td>25</td>
    </tr>
</table>
```

---

2. Creating a Form (`<form>`)

Definition:

A form in HTML allows users to **input and submit data** to a web server. The `<form>` element wraps input fields, and the data is sent when the user clicks the **submit button**.

Example Form Code:

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

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

    <label for="message">Message:</label>
    <textarea id="message" name="message"></textarea>

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

Form Elements and Definitions:

These are the tags and attributes used to make a fully functional form which post the data to webserver

Tag/Attribute	Definition	Example Usage
`<form>`	Defines a form for user input	`<form action="/submit" method="POST"> ... </form>`
`action`	Specifies where the form data is sent	`action="/submit"`
`method="POST"`	Sends data securely (alternative: `GET` for URL parameters)	`<form method="POST">`
`<label>`	Describes the input field (improves accessibility)	`<label for="name">Name:</label>`
`<input>`	Defines an input field	`<input type="text" name="name">`
`type="text"`	Allows text input	`<input type="text">`
`type="email"`	Ensures valid email input	`<input type="email">`
`type="password"`	Masks user input for security	`<input type="password">`
`type="checkbox"`	Creates a selectable checkbox	`<input type="checkbox">`
`type="radio"`	Creates radio buttons for selecting one option	`<input type="radio" name="gender" value="male">`
`type="radio"`	Creates radio buttons for selecting one option	`<input type="radio" name="gender" value="male">`
`<textarea>`	Allows multi-line text input	`<textarea name="message"></textarea>`
`<button>`	A clickable button (usually for form submission)	`<button type="submit">Submit</button>`
`required`	Makes the input field mandatory	`<input type="text" required>`
`placeholder`	Provides a hint in an input field	`<input type="text" placeholder="Enter Name">`

---

3. Example: Advanced Form with More Input Fields

```
<form action="/submit" method="POST">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
```

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

    <label for="hobbies">Choose Hobbies:</label>
    <input type="checkbox" name="hobbies" value="reading"> Reading
    <input type="checkbox" name="hobbies" value="sports"> Sports

    <label for="country">Select Country:</label>
    <select id="country" name="country">
        <option value="usa">USA</option>
        <option value="uk">UK</option>
        <option value="india">India</option>
    </select>

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

Description of the Form:

This is a **user registration form** that collects basic user information and preferences.

- **Action & Method:** The form submits data to `/submit` using the **POST** method.
- **Username & Password:** Requires users to enter a **username** and **password**.
- **Gender Selection:** Users can select their **gender** using **radio buttons** (only one option can be selected).
- **Hobbies Selection:** Users can select multiple hobbies using **checkboxes**.
- **Country Selection:** Users choose their country from a **dropdown (`<select>`)** menu.
- **Submit Button:** Registers the user by sending the entered data to the server.

Tips and Tricks:

1. HTML Learning: https://developer.mozilla.org/en-US/docs/Web/HTML
2. HTML Practice: https://www.naukri.com/code360/web-development

---

Follow tech blogs like Smashing Magazine, and Dev.to to stay current with industry trends