Table of Contents

Unit 1: Introduction to HTML

1.1 What is HTML?

- **HTML (HyperText Markup Language)** is the standard language used to create web pages.
- It tells the browser **what content to display** and how it's **structured**.
- HTML uses **tags** to define elements (e.g., <p>, <a>, <img>).

Note: *HTML is not a programming language; it is a markup language.*

1.2 History & Versions of HTML

Version	Year	Key Features
HTML 1.0	1993	Basic text & image formatting
HTML 3.2	1997	Tables, scripting support added
HTML 4.01	1999	CSS support, improved structure
XHTML 1.0	2000	XML-based strict rules
HTML5	**2014**	Video, audio, canvas, semantic tags, APIs introduced

1.3 Structure of an HTML Document

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My First HTML Page</title>
</head>
<body>
  <h1>Hello World!</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

Tag	Purpose
<!DOCTYPE>	Declares HTML version (HTML5)
<html>	Root element of the HTML page
<head>	Contains metadata, title, links, etc.
<meta>	Defines character set and other info
<title>	Title shown in browser tab
<body>	Contains all visible content on the page

1.4 Understanding Tags and Elements

- **Tags** are the keywords in angle brackets (< >).
- **Elements** are everything from the opening tag to the closing tag.

This is a paragraph.

Type	Example
Paired tags	
Self-closing tags	 , <img />

1.5 Comments in HTML

Used to add notes without displaying in the browser:

<!-- This is a comment -->

1.6 Basic Page Setup Template (Boilerplate)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Page Title</title>
</head>
<body>
  <h1>Main Heading</h1>
  <p>Page content goes here...</p>
</body>
</html>
```

1.7 Practice Exercises

1. Create a basic HTML page with:
    - A heading (<h1>)
    - A paragraph ( )
    - Page title in the browser tab
2. Add a comment inside your page.
3. Use the boilerplate template to start your first mini project.

Summary of Key Concepts

Concept	Description
HTML	Markup language for structuring content
Tags	Elements enclosed in < >
HTML5	Latest version with rich multimedia support
Boilerplate	Basic setup for any HTML document
Comments	Non-rendered notes for developers

Unit 2: Basic HTML Elements

2.1 Headings (<h1> to <h6>)

- HTML provides six levels of headings.
- <h1> is the most important, <h6> the least.

```
<h1>Main Heading</h1>
<h2>Subheading</h2>
<h3>Smaller Subheading</h3>
```

Best Practice: Use headings hierarchically (don't skip levels randomly).

2.2 Paragraphs and Line Breaks

- **Paragraph**: <p> tag defines a block of text.

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

- **Line Break**:
 tag inserts a single line break.

```
<p>This is<br>a new line.</p>
```

- **Horizontal Rule**: <hr> inserts a thematic break.

```
<hr>
```

2.3 Text Formatting Tags

Tag	Purpose	Example Output
<b>	Bold (non-semantic)	**Bold**
****	Important (semantic bold)	**Important**
<i>	Italic (non-semantic)	*Italic*
****	Emphasized (semantic italic)	*Emphasized*
<u>	Underlined	Underlined
<mark>	Highlight text	Highlighted
<sup>	Superscript	102 = 100
<sub>	Subscript	H2O
<code>	Code-style text	<code>let x = 5;</code>

2.4 Comments and Whitespace

- Comments: <!-- This is a comment -->
- Browsers ignore multiple spaces and line breaks unless specified with
, , or CSS.

2.5 HTML Entities

Used to display reserved characters:

- < = <
- > = >
- & = &
-   = non-breaking space
- © = ©

2.6 **Block vs Inline Elements**

Feature	Block Elements	Inline Elements
Display behavior	Start on a new line	Stay within the same line
Takes up	Full width of parent	Only as much width as needed
Nesting	Can contain other block or inline tags	Can contain only inline content
Common examples	, <p>, <h1> to <h6>,	, <a>, <b>, <i>, <img>

Example:

```
<!-- Block Element Example -->
<p>This is a block-level element.</p>

<!-- Inline Element Example -->
<p>This is <strong>inline bold text</strong> inside a paragraph.</p>
```

Tip: Use for generic block containers and for generic inline containers.

2.7 Practice Exercises

1. Create a page with:
    - All 6 heading tags
    - Two paragraphs and a horizontal rule
    - Inline formatting: bold, italic, underlined
2. Use
, <hr>, and HTML entities like ©, <, >
3. Demonstrate both block and inline elements inside a single section.

Summary of Key Concepts

Concept	Description
Headings	<h1> to <h6> define title/subtitle levels
Paragraph	<p> is used for text blocks
Line Break	 forces a new line
Formatting Tags	<b>, <i>, , etc. modify text style
Block vs Inline	Block = new line, Inline = inline display
HTML Entities	Special characters using &name; format

Unit 3: Links and Media in HTML

3.1 Anchor Tags (<a>) – Creating Links

Definition:

The <a> tag (anchor) is used to define hyperlinks, which can link to other webpages, sections within a page, or downloadable content.

Basic Syntax:

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

Common Attributes:

Attribute	Description
href	URL of the page or resource to link to
target	Specifies where to open the link (_blank, _self)
title	Tooltip text on hover
download	Instructs browser to download the file
rel	Describes relationship (noopener, noreferrer, nofollow)

Example 1: External Link with target and title

```
<a href="https://openai.com" target="_blank" title="Visit OpenAI's Homepage">OpenAI</a>
```

Example 2: Downloadable PDF with download

```
<a href="assets/html-cheatsheet.pdf" download="HTML-CheatSheet">Download HTML Cheat Sheet</a>
```

3.2 Image Tag (<img>) – Adding Images

Definition:

The <img> tag embeds images into a page. It's **self-closing** and requires at least src and alt attributes.

Basic Syntax:

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

Common Attributes:

Attribute	Description
src	Path or URL to the image
alt	Alternate text (essential for accessibility)
width	Width in px or %
height	Height in px or %
title	Tooltip text on hover
loading	Controls lazy loading (lazy, eager, auto)
referrerpolicy	How referrer info is passed with request

Example 1: Responsive Image

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

Example 2: External Image with Lazy Load

```
<img src="https://picsum.photos/300" alt="Random Image" loading="lazy" title="Random
Placeholder Image">
```

💡 **Use loading="lazy"** for performance optimization on scroll-heavy pages.

3.3 Video Tag (<video>) – Embedding Videos

Definition:

The <video> tag embeds video files into HTML. It allows playback control, streaming, and interaction.

Basic Syntax:

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

Common Attributes:

Attribute	Description
src	Source of the video (optional if using <source>)
controls	Shows play, pause, volume controls
autoplay	Starts playing automatically (must be muted)
muted	Mutes video by default (often used with autoplay)
loop	Repeats the video in a loop
poster	Image shown before the video starts

preload	How much video to preload (auto, metadata, none)

Example 1: Simple Video Player

```
<video src="intro.mp4" controls width="400"></video>
```

Example 2: Autoplaying Loop with Poster Image

```
<video autoplay muted loop poster="thumbnail.jpg" preload="auto" width="400">
  <source src="promo.mp4" type="video/mp4">
  Your browser does not support the video tag.
</video>
```

⚠ Autoplay **requires muted** due to browser restrictions.

3.4 Audio Tag (<audio>) – Embedding Audio

Definition:

The <audio> tag is used to embed audio files with optional playback controls.

Basic Syntax:

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

Common Attributes:

Attribute	Description
src	Path or URL to the audio (can also use <source>)
controls	Shows play/pause UI
autoplay	Plays audio when the page loads (usually muted)
loop	Repeats audio continuously
muted	Starts muted
preload	Defines loading behavior (none, auto, metadata)

Example 1: Audio Player

```
<audio src="background.mp3" controls></audio>
```

Example 2: Autoplay Looping Audio

```
<audio autoplay muted loop preload="auto">
  <source src="theme.mp3" type="audio/mpeg">
</audio>
```

3.5 Relative vs Absolute URLs

Type	Example	Use Case
Relative	images/photo.jpg, about.html	Refers to files inside the same project
Absolute	https://example.com/images/photo.jpg	External resources or full URLs

Example 1: Relative URL

```
<a href="contact.html">Contact Us</a>
<img src="images/logo.png" alt="Logo">
```

Example 2: Absolute URL

```
<a href="https://github.com" target="_blank">GitHub</a>
<img src="https://picsum.photos/200" alt="External Image">
```

3.6 Practice Tasks

1. Create a profile page with:
   - Name and image using <img>
   - Downloadable resume link with <a download>
   - Links to your GitHub & LinkedIn (with target="_blank")
2. Embed:
   - One video with controls and poster image
   - One background music loop (muted autoplay)
3. Use relative URLs for local files and absolute URLs for external ones.

Summary

Tag	Description	Must-Know Attributes
<a>	Creates a hyperlink	href, target, title, download, rel
<img>	Embeds an image	src, alt, title, width, loading
<video>	Embeds a video player	src, controls, autoplay, poster, loop
<audio>	Embeds an audio player	src, controls, autoplay, loop, muted

Unit 4: Lists and Tables in HTML

4.1 HTML Lists

HTML provides three types of lists:

1. **Ordered List** (<ol>)
2. **Unordered List** (<ul>)
3. **Description List** (<dl>)

4.1.1 Ordered List (<ol>)

Displays list items in a **numbered** sequence.

Basic Syntax:

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

Common Attributes:

Attribute	Description
type	Sets numbering style: 1, A, a, I, i
start	Starts numbering from a custom number
reversed	Displays items in reverse order

Example 1: Default Ordered List

```
<ol>
  <li>Wake up</li>
  <li>Brush teeth</li>
</ol>
```

Example 2: Alphabetical Reversed List

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

4.1.2 Unordered List (<ul>)

Displays list items in **bullet points**.

Basic Syntax:

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

Example: Shopping List

```
<ul>
  <li>Milk</li>
  <li>Bread</li>
  <li>Eggs</li>
</ul>
```

Bullet styles can be customized using CSS (disc, circle, square).

4.1.3 Description List (<dl>)

Used for **term–description** pairs (like glossaries or FAQs).

Basic Syntax:

```
<dl>
  <dt>HTML</dt>
  <dd>HyperText Markup Language</dd>
</dl>
```

Example: Tech Glossary

```
<dl>
  <dt>CSS</dt>
  <dd>Used for styling HTML elements</dd>
  <dt>JS</dt>
  <dd>Used to add interactivity</dd>
</dl>
```

4.2 HTML Tables

Tables are used to display tabular data in rows and columns.

4.2.1 Basic Table Structure

Basic Syntax:

```
<table>
  <tr>
    <th>Heading 1</th>
    <th>Heading 2</th>
  </tr>
  <tr>
    <td>Row 1, Cell 1</td>
    <td>Row 1, Cell 2</td>
  </tr>
</table>
```

Tag	Description
<table>	Main container
<tr>	Table row
<th>	Table header cell (bold + centered)
<td>	Table data cell

4.2.2 Table Attributes (HTML5)

⚠ Visual styling should be done with **CSS**, not deprecated attributes like border, bgcolor, etc.

Attribute	Description
colspan	Merges cells horizontally across columns
rowspan	Merges cells vertically across rows
scope	Improves accessibility (row, col)

Example 1: Basic Table with Headers

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Alice</td>
    <td>22</td>
  </tr>
  <tr>
    <td>Bob</td>
    <td>25</td>
  </tr>
</table>
```

Example 2: Merged Table with colspan and rowspan

```
<table border="1">
  <tr>
    <th rowspan="2">Name</th>
    <th colspan="2">Marks</th>
  </tr>
  <tr>
    <th>Math</th>
    <th>Science</th>
  </tr>
  <tr>
    <td>Alex</td>
    <td>90</td>
    <td>88</td>
  </tr>
</table>
```

Good for displaying exam results or comparison tables.

4.3 Practice Tasks

1. Create:
    - A shopping list using
    - A numbered to-do list using <ol type="I">
    - A glossary using <dl>, <dt>, and <dd>
2. Build a table showing student marks with:
    - Headers
    - Rowspan for names
    - Colspan for merged headers
3. Convert a static syllabus into a clean table format

Summary

Feature	Element(s) Used	Notes
Unordered List	,	Bullet points
Ordered List	,	Numbered items, customizable with type
Description List	<dl>, <dt>, <dd>	Term-definition pairs
Table Structure	<table>, <tr>, <th>, <td>	Used to format tabular data
Merging Cells	colspan, rowspan	Merges cells horizontally/vertically

Unit 5: Forms and Input Elements in HTML

5.1 What is a Form in HTML?

Definition:

A **form** is a section of an HTML document that contains **interactive controls** for users to submit data to a server or handle with JavaScript.

Basic Syntax:

```
<form action="submit.php" method="post">
  <!-- form controls go here -->
</form>
```

Attribute	Description
action	URL to which the form data will be submitted
method	HTTP method (get or post) to send data
target	Where to display the response (_self, _blank, etc.)
autocomplete	Enables or disables browser auto-completion (on, off)

GET = URL-based (visible), for searches
POST = Body-based (secure), for sensitive data

5.2 Basic Input Elements (<input>)

The <input> tag creates **form fields**, and the type attribute defines the input behavior.

5.2.1 Common Input Types

Type	Description	Example
text	Single-line text	<input type="text">
password	Hides input	<input type="password">
email	Validates email format	<input type="email">
number	Numeric input	<input type="number">
date	Date picker	<input type="date">
checkbox	Multiple selections	<input type="checkbox">
radio	Single selection within a group	<input type="radio" name="gender">
submit	Submit button	<input type="submit" value="Send">
reset	Resets form fields	<input type="reset" value="Clear">
file	Upload files	<input type="file">

Example 1: Basic Input Form

```
<form>
  <label for="username">Name:</label>
  <input type="text" id="username" name="username">
  <br><br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email">
  <br><br>
  <input type="submit" value="Submit">
</form>
```

5.3 Labels and Placeholders

- <label>: Describes the input field (improves accessibility)
- for attribute links the label to the input's id
- placeholder gives a hint inside the input field

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

5.4 Radio Buttons and Checkboxes

Radio Buttons: Choose one option only

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

Checkboxes: Select multiple options

```
<p>Choose your skills:</p>
<input type="checkbox" name="skills" value="html"> HTML
<input type="checkbox" name="skills" value="css"> CSS
<input type="checkbox" name="skills" value="js"> JavaScript
```

Radio buttons must share the same name to work as a group.

5.5 Select Box and Dropdown Menus

Basic Syntax:

```
<select name="country">
  <option value="india">India</option>
  <option value="usa">USA</option>
</select>
```

Attributes:

Attribute	Description
selected	Sets the default selected option
multiple	Allows multiple selections
size	Number of options visible at once

5.6 Text Areas

Used for **multi-line input** like feedback or comments.

```
<label for="message">Your Message:</label><br>
<textarea id="message" name="message" rows="4" cols="50">Enter your text
here...</textarea>
```

5.7 Fieldset and Legend

Used to **group form elements** together with a title.

```
<fieldset>
  <legend>Personal Info</legend>
  <label>Name:</label>
  <input type="text">
</fieldset>
```

5.8 Form Validation Attributes

Attribute	Description
required	Field must be filled before submitting
min, max	Minimum or maximum value (for number, date)
maxlength	Max character limit for input text
pattern	Regex-based custom validation (e.g., phone)
readonly	Makes field uneditable
disabled	Field cannot be interacted with

Example: Required Email with Pattern

<input type="email" required pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$">

5.9 Practice Tasks

1. Create a **sign-up form** with:
   - Name, Email, Password, Gender, Skills
   - Validation using required, pattern, and maxlength
2. Create a **feedback form** using <textarea> and <fieldset>
3. Use a **dropdown** with multiple options (countries or states)
4. Add a **file upload** and a submit button

Summary

Element	Purpose	Common Attributes
<form>	Wraps all input fields	action, method, autocomplete
<input>	Creates form controls	type, name, placeholder, required
<label>	Associates text with input field	for (must match input id)
<select>	Dropdown list	multiple, size, selected
<textarea>	Multiline text input	rows, cols, placeholder
<fieldset>	Groups inputs	Used with <legend>

Unit 6: Semantic HTML

6.1 What is Semantic HTML?

Definition:

Semantic HTML refers to using HTML tags that **clearly describe their meaning and purpose** both to the browser and to developers (and assistive technologies like screen readers).

Helps with:

- SEO (Search Engine Optimization)
- Accessibility (screen readers, keyboard nav)
- Maintainability & clean code

6.2 Non-Semantic vs Semantic Tags

Type	Example Tags	Meaning
Non-Semantic	,	No inherent meaning (generic containers)
Semantic	<header>, <footer>, <nav>	Clearly describe content purpose

6.3 Common Semantic Tags

Tag	Purpose
<header>	Introduces a page or section (often contains logo, nav, h1)
<nav>	Contains navigation links
<main>	Main content of the document (unique per page)
<section>	Thematic grouping of content (e.g., features, testimonials)
<article>	Self-contained content (e.g., blog post, comment, card)
<aside>	Sidebar or related content (ads, tips, links)
<footer>	Bottom of page/section (copyright, links, contact)
<figure>	Wraps media (like image) with a caption
<figcaption>	Provides caption for <figure>
<mark>	Highlights important content
<time>	Represents date/time (for events, articles, etc.)
<address>	Provides contact details

6.4 Tag-by-Tag Usage Examples

<header> + <nav> Example:

```
<header>
  <h1>My Website</h1>
  <nav>
    <a href="#home">Home</a> |
    <a href="#services">Services</a> |
    <a href="#contact">Contact</a>
  </nav>
</header>
```

<main> + <section> Example:

```
<main>
  <section id="about">
    <h2>About Us</h2>
    <p>We build amazing web apps.</p>
  </section>

  <section id="services">
    <h2>Our Services</h2>
    <p>Web Development, UI/UX Design, SEO</p>
  </section>
</main>
```

<article> + <aside> Example:

```
<article>
  <h2>Top 5 HTML Tips</h2>
  <p>Learn how to write clean and modern HTML...</p>
</article>

<aside>
  <h3>Related Posts</h3>
  <ul>
    <li><a href="#">HTML5 Forms</a></li>
    <li><a href="#">CSS Flexbox Guide</a></li>
  </ul>
</aside>
```

<figure> + <figcaption> Example:

```
<figure>
  <img src="team.jpg" alt="Team photo">
  <figcaption>Our awesome team at the 2024 Hackathon</figcaption>
</figure>
```

<footer> Example:

```
<footer>
  <p>&copy; 2025 MyWebsite. All rights reserved.</p>
  <address>Contact us: hello@mywebsite.com</address>
</footer>
```

<mark>, <time>, <address>:

```
<p>Don't forget to attend the <mark>HTML Workshop</mark> on <time datetime="2025-08-
10">August 10</time>.</p>

<address>
  123 Web Lane,<br>
  Bangalore, India
</address>
```

6.5 Practice Tasks

1. Create a **complete webpage layout** using:
   - <header>, <nav>, <main>, <section>, <article>, <footer>
2. Use <figure> and <figcaption> to describe an image.
3. Add a date with <time>, a tip box using <aside>, and highlighted text with <mark>.
4. Replace with semantic equivalents in an old layout.

Summary

Semantic Tag	Purpose
<header>	Top banner, title, and nav links
<nav>	Contains site navigation
<main>	Core content of the page
<section>	A group of related content with a common theme
<article>	Independent item (blog post, card, comment)
<aside>	Supplementary content (sidebar, tips)
<footer>	End of section or page (contact, credits)
<figure>	Groups media with caption
<time>	Machine-readable date/time
<address>	Contact info, typically used in footer

Unit 7: HTML5 APIs

7.1 What Are HTML5 APIs?

Definition:

HTML5 APIs are built-in browser interfaces that allow web pages to interact with the device, browser environment, or user — without needing plugins.

HTML5 APIs = JavaScript + HTML + Browser support
Enables rich, dynamic, and interactive web applications

7.2 Common HTML5 APIs

API Name	Purpose
Geolocation API	Get user's location
Web Storage API	Store data in the browser (local/session)
Drag and Drop API	Drag and move elements interactively
Canvas API	Draw graphics and animations
Audio/Video API	Programmatically control media elements
History API	Manipulate browser history
Web Workers API	Run scripts in background threads (multi-threading)
Fullscreen API	Display elements in fullscreen

7.3 Web Storage API

Allows storing data **locally** in the user's browser — persists across sessions (localStorage) or tabs (sessionStorage).

localStorage

```
// Store
localStorage.setItem("name", "Alice");

// Retrieve
let user = localStorage.getItem("name");

// Delete
localStorage.removeItem("name");
```

Persists even after browser is closed

sessionStorage

```
sessionStorage.setItem("sessionId", "xyz123");
let session = sessionStorage.getItem("sessionId");
```

Cleared when the tab or window is closed

7.4 Geolocation API

Fetches user's current geographic location (requires user permission).

```
navigator.geolocation.getCurrentPosition(
  function(position) {
    console.log("Latitude: " + position.coords.latitude);
    console.log("Longitude: " + position.coords.longitude);
  },
  function(error) {
    console.error("Error: ", error.message);
  }
);
```

Method	Description
getCurrentPosition()	Gets the user's current location
watchPosition()	Tracks location continuously

Use HTTPS and handle errors (e.g., permission denied)

7.5 Drag and Drop API

Enables users to drag elements and drop them into targets.

```
<div id="drag1" draggable="true" ondragstart="drag(event)">Drag Me</div>
<div id="dropzone" ondrop="drop(event)" ondragover="allowDrop(event)">Drop Here</div>
function allowDrop(ev) {
  ev.preventDefault();
}
function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}
function drop(ev) {
  ev.preventDefault();
  let data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
```

7.6 Canvas API

Allows drawing 2D graphics, charts, or games using JavaScript.

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
black;"></canvas>

<script>
  const c = document.getElementById("myCanvas");
  const ctx = c.getContext("2d");
  ctx.fillStyle = "green";
  ctx.fillRect(10, 10, 150, 75);
</script>
```

Method	Purpose
getContext("2d")	Access the 2D drawing context
fillRect(x, y, w, h)	Draw a filled rectangle
arc(), lineTo()	Draw shapes

7.7 Audio and Video API (Programmatic Control)

You can control <audio> or <video> elements using JavaScript.

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

<button onclick="document.getElementById('myVideo').play()">Play</button>
```

JavaScript Method	Description
.play()	Starts playback
.pause()	Pauses media
.volume = 0.5	Sets volume level
.currentTime	Seeks to a time

7.8 History API

Modify browser history without reloading the page (used in SPAs like React).

history.pushState({ page: 1 }, "title", "?page=1");

Method	Purpose
pushState()	Add a state to browser history
replaceState()	Replace current history entry
onpopstate	Listen to back/forward navigation

7.9 Fullscreen API

Use JavaScript to enter fullscreen mode for an element.

```
<button onclick="openFullscreen()">Go Fullscreen</button>

<script>
function openFullscreen() {
  document.documentElement.requestFullscreen();
}
</script>
```

Useful for games, presentations, or media players.

7.10 Practice Tasks

1. Create a **Geo-tracker** that prints the user's latitude and longitude.
2. Use **localStorage** to store form inputs (like name/email).
3. Create a **Drag & Drop** task list.
4. Use **Canvas** to draw a traffic light or flag.
5. Build a **custom video player** with play, pause, and volume controls.
6. Add buttons that use the **History API** to change URLs without reload.

Summary

API	Purpose	How to Use
localStorage	Store persistent data in the browser	localStorage.setItem()
geolocation	Get user's current location	navigator.geolocation.getCurrentPosition()
drag/drop	Enable interactive movement	dragstart, drop, ondragover
canvas	Draw graphics	canvas.getContext("2d")
video/audio	Programmatic media control	.play(), .pause()
history	Modify browser history dynamically	pushState(), replaceState()
fullscreen	Launch full screen on elements	element.requestFullscreen()

Unit 8: Advanced HTML Features

Topics Covered:

- Meta Tags (extended)
- data-* attributes
- <iframe> tag
- <svg> for scalable graphics
- <math> using MathML
- Web Components (intro)

8.1 Meta Tags (Recap + Extras)

Besides <meta charset> and <meta viewport>, HTML supports more <meta> tags useful for SEO, social sharing, and browser behavior.

Common Meta Tags

Meta Tag	Purpose
<meta charset="UTF-8">	Character encoding
<meta name="viewport" content="width=device-width, initial-scale=1.0">	Responsive behavior
<meta name="description" content="Page summary">	SEO snippet for search engines
<meta name="author" content="Your Name">	Author info
<meta name="keywords" content="html, web, tags">	Keywords (deprecated for Google)
<meta http-equiv="refresh" content="5;url=https://example.com">	Auto-redirect after delay

8.2 data-* Attributes

Definition:

HTML5 allows embedding custom data in any element using data-* attributes. Useful for storing metadata accessible via JavaScript.

Syntax:

<button data-user-id="42" data-role="admin">Edit User</button>

Access via JavaScript:

```
const btn = document.querySelector('button');
console.log(btn.dataset.userId); // "42"
console.log(btn.dataset.role);   // "admin"
```

Use data-* for extra data needed by JS without cluttering the DOM.

8.3 <iframe> – Embedding Web Content

Definition:

The <iframe> tag allows you to embed **another HTML page, YouTube video, PDF, or external resource** within your page.

Syntax:

```
<iframe src="https://www.wikipedia.org" width="600" height="400"
title="Wikipedia"></iframe>
```

Attribute	Purpose
src	URL to embed
width/height	Frame size
title	Accessible label
allowfullscreen	Enables full-screen mode (for videos)
loading="lazy"	Defers loading until visible (performance)

⚠ Some websites block iframe embedding using X-Frame-Options.

8.4 <svg> – Scalable Vector Graphics

Definition:

<svg> is used to define vector graphics like shapes, icons, and charts that are **scalable without quality loss**.

Example:

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" fill="skyblue" stroke="black" stroke-width="2" />
</svg>
```

SVG Element	Meaning
<circle>	Draws a circle
<rect>	Rectangle
<line>	Straight line
<text>	Display text
<path>	Complex shapes

 SVG is ideal for responsive logos, icons, charts.

8.5 <math> – MathML for Mathematical Markup

Definition:

<math> is part of MathML, a markup language for displaying **mathematical notations** in browsers.

Example:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <msup>
    <mi>x</mi>
    <mn>2</mn>
  </msup>
</math>
```

➡ This shows "x^2" in a math-friendly format.

📌 Not fully supported in all browsers yet. Common in scientific/academic content.

8.6 Introduction to Web Components

Definition:

Web Components are a set of modern browser APIs that allow you to create **reusable, encapsulated HTML elements**.

Key Concepts:

Concept	Explanation
Custom Elements	Define your own HTML tags (e.g., <my-button>)
Shadow DOM	Isolates styles and scripts within a component
HTML Templates	Reusable chunks of markup

Minimal Example:

```
<template id="card-template">
  <style>
    .card { border: 1px solid #ccc; padding: 1rem; }
  </style>
  <div class="card">
    <slot></slot>
  </div>
</template>

<script>
  class MyCard extends HTMLElement {
    constructor() {
      super();
      const template = document.getElementById('card-template').content;
      this.attachShadow({ mode: 'open' }).appendChild(template.cloneNode(true));
    }
```

```
  customElements.define('my-card', MyCard);
</script>

<my-card>Hello in a custom card!</my-card>
```

Web Components are ideal for building reusable design systems.

8.7 Practice Tasks

1. Add a data-* attribute to a product card and print it using JS.
2. Embed a YouTube video or Google Form using <iframe>.
3. Draw a shape using SVG: circle, rectangle, or pie chart.
4. Write a mathematical expression using <math> for a formula.
5. Create a custom HTML element using template + Shadow DOM.

Summary

Feature	Tag / Concept	Purpose
Custom Data	data-*	Store reusable custom attributes
Embedding	<iframe>	Embed other pages or media
Vector Graphics	<svg>	Scalable shapes and icons
Math Markup	<math>	Render equations and formulas
Web Components	<template>, Shadow DOM	Reusable custom HTML components

Unit 9: Accessibility & Best Practices

9.1 What is Web Accessibility?

Definition:

Web Accessibility (often abbreviated as **a11y**) ensures that websites are usable by **everyone**, including users with:

- Vision impairments (blind, color-blind)
- Hearing impairments
- Motor difficulties (keyboard-only users)
- Cognitive limitations

Accessibility is not just ethics — it's often a legal requirement.

9.2 Semantic HTML & Accessibility

Semantic HTML helps assistive technologies like screen readers **understand your layout**.

Use Tags Like:

Tag	Purpose
<nav>	Navigation links
<main>	Main content of the page
<section>	Logical content groups
<article>	Standalone content
<aside>	Complementary information
<footer>	Page or section footer

9.3 alt Attributes for Images

Always add meaningful alt text:

```
<img src="team.jpg" alt="Our product team working on a whiteboard">
```

Case	What to Do
Informative image	Use descriptive alt
Decorative image	Use alt=""
Image contains text	Describe the visible text in alt
Image is a logo	Use alt like "Company Logo"

Don't leave alt blank unless the image is purely decorative.

9.4 Using <label> with Form Inputs

Labels make inputs accessible to screen readers and users navigating with keyboards.

Correct:

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

🔗 The for attribute must match the input's id.

9.5 Keyboard Navigation

Websites should be fully usable via the **Tab** and **Enter** keys.

Element	Keyboard Focusable?
<a href>	Yes
<button>	Yes
<input>, <select>, <textarea>	Yes
** **	No (unless tabindex is added)

Example with tabindex:

```
<div tabindex="0">Focusable Div</div>
```

tabindex="0" makes non-focusable elements keyboard-accessible
Avoid tabindex > 1 — it can mess up focus order

9.6 ARIA Attributes (Accessible Rich Internet Applications)

ARIA attributes help **describe roles, states, and properties** to assistive technologies.

Attribute	Description
role="button"	Declares element as a button
aria-label	Provides an accessible label
aria-hidden="true"	Hides element from screen readers
aria-expanded	Shows toggle status of elements

Example:

```
<div role="button" tabindex="0" aria-label="Close modal">X</div>
```

⚠ Use ARIA only when semantic HTML doesn't suffice.

9.7 Color Contrast & Readability

- Use **high contrast** between text and background
- Avoid conveying information **only by color**

Good contrast:

color: #000; background-color: #fff;

Bad contrast:

color: #999; background-color: #ccc;

9.8 Practice Tasks

1. Write an accessible form with <label>, required, and aria-*
2. Add alt text to 5 meaningful images
3. Use tabindex="0" to make a focusable
4. Use role="button" on a clickable and test keyboard activation
5. Fix color contrast issues in a given page

Summary

Feature	Tag / Attribute	Purpose
Alt Text	alt="..."	Describe images for screen readers
Label/Form	<label for="">	Connects inputs with descriptions
Keyboard Nav	tabindex, buttons	Enables full keyboard usage
ARIA Roles	role, aria-label	Extra info for non-semantic UI
Semantic Layout	<main>, <nav>	Improves structure & a11y
Visual Contrast	CSS colors	Improves readability

Unit 10: HTML + CSS/JS Integration

10.1 Overview

In a real web project:

- **HTML** = structure (the "bones")
- **CSS** = style (the "clothes")
- **JavaScript (JS)** = behavior (the "brain")

HTML alone is static. CSS and JS bring it to life visually and functionally.

10.2 Adding CSS to HTML

There are 3 ways to apply CSS to an HTML page:

1. Inline CSS (inside an element)

```
<p style="color: green; font-size: 20px;">Hello</p>
```

Not recommended for larger projects (hard to maintain)

2. Internal CSS (inside <style> in the <head>)

```
<head>
  <style>
    h1 {
      color: blue;
      text-align: center;
    }
  </style>
</head>
```

3. External CSS (best practice)

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

In styles.css:

```
body {
  background-color: #f0f0f0;
  font-family: Arial, sans-serif;
}
```

Keeps code organized and reusable

10.3 Adding JavaScript to HTML

1. Inline JS (not recommended)

```
<button onclick="alert('Hello')">Click Me</button>
```

2. Internal JS (in <script> tag)

```
<script>
  function greet() {
    alert("Welcome!");
  }
</script>

<button onclick="greet()">Click</button>
```

3. External JS (best practice)

In HTML:

```
<script src="script.js"></script>
```

In script.js:

```
console.log("JS file connected!");
```

Keeps logic separate and clean

10.4 Accessing the DOM (Document Object Model)

The DOM is a JS representation of the HTML structure — allowing JavaScript to read or change HTML content dynamically.

Example:

HTML:

```
<p id="msg">Welcome</p>
<button onclick="changeText()">Change</button>
```

JS:

```
function changeText() {
  document.getElementById("msg").innerText = "Hello, world!";
}
```

getElementById() is a common way to access elements

10.5 JavaScript Events

JavaScript can respond to **events** like:

Event	Trigger
onclick	When an element is clicked
onmouseover	When mouse hovers
onchange	Input field value changes
onload	Page fully loads

Example:

```
<input type="text" id="name" onchange="sayHello()">

<script>
  function sayHello() {
    const val = document.getElementById("name").value;
    alert("Hello " + val);
  }
</script>
```

10.6 Connecting HTML, CSS, JS in One Project

Project Skeleton:

```
project/
│
├── index.html
├── styles.css
└── script.js
```

index.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Page</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1 id="title">Hello</h1>
  <button onclick="updateText()">Click</button>

  <script src="script.js"></script>
</body>
</html>
```

styles.css:

```
body {
  text-align: center;
  background-color: lightyellow;
}
```

script.js:

```
function updateText() {
  document.getElementById("title").innerText = "Updated!";
}
```

10.7 Single Page Application (SPA) Overview (Intro)

SPA is a web application that loads a single HTML page and updates content dynamically without reloading.

Popular SPA frameworks: React, Angular, Vue

Without SPA:

Clicking a link → reloads entire page

With SPA:

Clicking a link → only changes part of the page (using JS + routing)

10.8 Practice Tasks

1. Add external CSS and JS files to a simple HTML page.
2. Create a button that changes a paragraph's color using JS.
3. Use getElementById and innerText to modify DOM elements.
4. Make a form where clicking "Submit" shows the name via alert.
5. Create a layout using internal CSS and hook a script that hides an element.

Summary

Integration Type	Method	Recommended?
CSS - Inline	style="..."	No
CSS - Internal	<style> in <head>	Yes (for small projects)
CSS - External	<link href="...">	✓Best
JS - Inline	onclick="..."	No
JS - Internal	<script> in HTML	Yes
JS - External	<script src="...">	✓Best

Unit 11: DevTools & Debugging

11.1 What are Developer Tools?

Definition:

DevTools are built-in browser tools that help developers **inspect, debug, and optimize** webpages.

Available in most modern browsers (Chrome, Firefox, Edge)
Shortcut: **F12** or **Right-click → Inspect**

11.2 Opening DevTools

Chrome Shortcuts:

Tool	Shortcut
Open DevTools	Ctrl + Shift + I or F12
Open Console	Ctrl + Shift + J
Inspect Element	Right-click → Inspect

11.3 Key Panels in Chrome DevTools

1. Elements Panel

- View & edit HTML structure in real-time
- Modify tags, attributes, classes, inline styles
- Highlights corresponding element on the page

Try:

```
<p id="demo">Hello</p>
```

- Right-click → Inspect → Change `<p>` content or color live

2. Styles Panel

- Shows CSS styles applied to selected element
- Allows live editing, adding, disabling CSS rules

Try changing:

- Font size
- Background color
- Adding `display: none`

3. Console Panel

- Shows errors, warnings, logs from JavaScript

- Allows you to run JS commands interactively

Example:

```
console.log("Hello DevTools");
document.getElementById("demo").style.color = "red";
```

4. Network Panel

- Monitors all network requests (HTML, CSS, JS, images, APIs)
- Use it to debug missing files or slow resources

Reload page while open to see:

- Request method (GET, POST)
- Status codes (200 OK, 404 Not Found)

5. Sources Panel

- View and debug JavaScript files
- Set breakpoints to pause and inspect JS step by step

6. Device Toolbar (Responsive View)

- Simulates different screen sizes & devices
- Useful for testing mobile responsiveness

Click 📱 icon or press Ctrl + Shift + M

11.4 Real Debugging Scenarios

Example: Fixing a Broken Element

```
<h1 class="title">Welcome</h1>
```

In CSS:

```
.title {
  color: ;
}
```

In DevTools:

- Go to **Elements**
- Select <h1 class="title">
- In **Styles**, you'll see the missing value
- Fix it live to color: blue;

Example: JavaScript Error

HTML:

```
<button onclick="sayHi()">Click Me</button>
```

JS:

```
function sayHi() {
  document.getElementById("msg").innerText = "Hi!";
}
```

But there's no element with ID msg.

In **Console**:

- Error: Cannot set property 'innerText' of null
- Fix: Add in HTML

11.5 Inspecting Forms

Use DevTools to:

- Check input name and type
- See if required or pattern is applied
- Test forms quickly without reloading

11.6 Using Lighthouse in DevTools

- Generate automated reports on **performance, accessibility, SEO**
- Go to → DevTools > Lighthouse → Click "Analyze page load"

Get real suggestions on:

- Improving load speed
- Fixing accessibility issues
- Enhancing mobile performance

11.7 Practice Tasks

1. Inspect and edit the text of any element using Elements panel
2. Use Console to change the color of an element
3. Simulate a mobile device using Device Toolbar
4. Find and fix a broken CSS rule using Styles panel
5. Reload a page and inspect the Network requests
6. Use Lighthouse to analyze a simple HTML page

Summary

Tool/Panel	Use Case
Elements	Live edit HTML structure
Styles	Debug and test CSS
Console	Run JS, view errors
Network	Monitor resource loading
Sources	Debug JavaScript with breakpoints
Device Toolbar	Simulate mobile viewports
Lighthouse	Performance, SEO, and a11y audits

Unit 12: Project-Based HTML Practice

12.1 Objectives

By the end of this unit, students will:

- Build complete static pages using only HTML (optionally with CSS)
- Strengthen understanding of semantic tags, structure, and form controls
- Gain confidence building professional-looking UI layouts
- Prepare components for personal portfolios and interviews

PROJECT 1: Personal Portfolio Homepage

Description:

A structured personal website showcasing About, Skills, Projects, and Contact.

Features:

- Navigation with anchors
- Use of <header>, <section>, <footer>
- Contact form with required fields

Sample:

```
<header>
  <h1>John Doe</h1>
  <nav>
    <a href="#about">About</a>
    <a href="#skills">Skills</a>
    <a href="#projects">Projects</a>
    <a href="#contact">Contact</a>
  </nav>
</header>

<section id="about">
  <h2>About Me</h2>
  <p>Aspiring front-end developer with a passion for design...</p>
</section>

<section id="skills">
  <h2>Skills</h2>
  <ul>
    <li>HTML5</li>
    <li>CSS3</li>
    <li>JavaScript</li>
  </ul>
</section>

<section id="projects">
  <h2>My Projects</h2>
  <article>
    <h3>To-Do App</h3>
```

```
    <p>Created using HTML, CSS, JS.</p>
  </article>
</section>

<section id="contact">
  <h2>Contact Me</h2>
  <form>
    <label>Name: <input type="text" required></label><br><br>
    <label>Email: <input type="email" required></label><br><br>
    <label>Message: <textarea></textarea></label><br><br>
    <input type="submit" value="Send">
  </form>
</section>

<footer>
  <p>© 2025 John Doe</p>
</footer>
```

PROJECT 2: Blog Post Layout

Description:

Create a simple blog post template using semantic layout and readable typography.

Features:

- Use of <article>, <aside>, and <figure>
- Display a cover image and post content

Sample:

```
<article>
  <h1>The Journey of Learning HTML</h1>
  <p><em>Published on July 10, 2025</em></p>

  <figure>
    <img src="blog.jpg" alt="HTML code on screen" width="100%">
    <figcaption>Learning HTML in 2025</figcaption>
  </figure>

  <p>HTML is the foundation of the web. It allows us to create structured
content...</p>
</article>

<aside>
  <h3>Author</h3>
  <p>Jane Developer is a tech writer who loves teaching HTML and CSS.</p>
</aside>
```

PROJECT 3: Resume / CV Page

Description:

Build a clean resume using lists, sections, and tables.

Features:

- Use of <table> or <ul> for listing skills/experience
- Include Contact, Education, Experience, Skills

Sample:

```
<header>
  <h1>Jane Doe</h1>
  <p>Email: jane@example.com | Location: India</p>
</header>

<section>
  <h2>Education</h2>
  <ul>
    <li>B.E. Computer Science - XYZ University (2021–2025)</li>
  </ul>
</section>

<section>
  <h2>Experience</h2>
  <ul>
    <li>Intern at TechCorp - Frontend Development (2024)</li>
  </ul>
</section>

<section>
  <h2>Skills</h2>
  <table border="1">
    <tr><th>Skill</th><th>Level</th></tr>
    <tr><td>HTML</td><td>Advanced</td></tr>
    <tr><td>CSS</td><td>Intermediate</td></tr>
  </table>
</section>
```

PROJECT 4: Contact Form Page

Description:

Create a full form page with validation fields, dropdowns, checkboxes, and submission.

Features:

- Input types: text, email, checkbox, radio
- required, minlength, placeholder, pattern

Sample:

```
<form>
  <h2>Contact Us</h2>
  <label>Full Name:
    <input type="text" name="name" required placeholder="Your full name">
  </label><br><br>

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

  <label>Subject:
    <select required>
      <option value="">Select</option>
      <option>Support</option>
      <option>Feedback</option>
      <option>General</option>
    </select>
  </label><br><br>
```

```
  <label>Message:
    <textarea required></textarea>
  </label><br><br>

  <label>
    <input type="checkbox" required> I accept the terms
  </label><br><br>

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

PROJECT 5: Weather UI Card (HTML only)

Description:

Design a weather card UI using only HTML and internal CSS.

Features:

- Use of icons/images
- Weather information: city, temperature, status

Sample:

```
<section style="width: 250px; border: 1px solid #ccc; padding: 1rem;">
  <h2>Weather in Chennai</h2>
  <img src="weather-icon.png" alt="Sunny" width="50">
  <p>Temperature: 34°C</p>
  <p>Status: Sunny</p>
</section>
```

12.2 Suggested Exercises

Project	Concepts Reinforced
Portfolio Page	Semantic layout, anchors, contact form
Blog Post	Articles, figures, sidebars
Resume Page	Lists, tables, headers
Contact Form	Forms, input types, validations
Weather Card	Internal CSS, images, layout

Summary

- Practice is the best way to **solidify theory**
- These projects are HTML-only but can be upgraded with CSS/JS later
- Ideal for **portfolios**, **assessments**, or **internship submissions**

Unit 13: Interview & Certification Preparation

This unit prepares students for **placements, internships, certifications**, and **HTML-focused interviews** by reviewing key concepts, practicing MCQs, and solving real-world problem statements.

13.1 HTML Interview Concepts Checklist

Before appearing for interviews or exams, students should master:

Category	Key Topics to Revise
Basic HTML Structure	<!DOCTYPE>, <html>, <head>, <body>
Semantic Tags	<header>, <main>, <article>, <footer>
Forms	<form>, input types, labels, attributes
Media	<img>, <video>, <audio>, <iframe>
Lists & Tables	, , <table>, <thead>, etc.
Links & Navigation	<a>, href, target, <nav>
HTML5 Features	Web Storage, Canvas, Geolocation
Accessibility	alt, label, tabindex, ARIA
Integration	Linking CSS/JS, DOM manipulation basics
Debugging & Best Practices	DevTools, SEO meta tags, code structure

---

13.2 Top HTML Interview Questions (with Answers)

Basic Level

1. **What is HTML?**
   - HTML stands for *HyperText Markup Language*. It is used to structure content on the web.
2. **What is the difference between and ?**
   - is a block-level container; is inline.
3. **What are void elements in HTML?**
   - Elements that do not have a closing tag, e.g.,
, <img>, <input>
4. **What does the alt attribute do?**
   - Provides alternate text for images; used by screen readers and when the image fails to load.
5. **Difference between id and class?**
   - id is unique; class can be reused across multiple elements.

Intermediate Level

6. **What is the use of the <fieldset> and <legend> tags?**
   - Used to group form elements and provide a title for that group.
7. **What is semantic HTML?**
   - HTML that clearly describes its meaning (like <article>, <section>), improving accessibility and SEO.
8. **What is the purpose of the target="_blank" attribute?**

- Opens the linked document in a new tab or window.

9. **How is HTML5 different from HTML4?**
   - HTML5 introduced semantic elements, audio/video support, form input types, and APIs like Canvas, Web Storage.
10. **What is ARIA and why is it used?**
    - ARIA (Accessible Rich Internet Applications) provides extra information to screen readers and assistive tools.

Advanced Level

11. **What are data-* attributes and how are they used?**
    - Custom attributes used to store extra data accessible via JS (element.dataset.name)
12. **How can you make a non-interactive element accessible via keyboard?**
    - Use tabindex="0" and ARIA roles if necessary.
13. **What is Shadow DOM?**
    - Part of Web Components; allows encapsulation of markup, style, and behavior inside custom elements.
14. **What are some ways to improve HTML performance?**
    - Minimize inline styles/scripts, defer JS, use lazy loading for images/iframes, optimize structure.
15. **How do screen readers interpret HTML structure?**
    - They rely on semantic tags, heading levels, alt text, labels, and ARIA roles for navigation and understanding.

13.3 MCQs: HTML Practice Test

#	Question	Options	Answer
1	Which tag is used to define an unordered list?	a) b) c) d) <list>	a
2	What does <!DOCTYPE html> do?	a) Adds a comment b) Defines version c) Declares HTML5	c
3	Which attribute is used for image alt text?	a) text b) alt c) title d) caption	b
4	What tag is used to embed video?	a) <vid> b) <movie> c) <video> d) <media>	c
5	Which input type is used for selecting a file?	a) text b) file c) image d) upload	b

✅ *Total MCQs provided: 30+* (You can request the full test bank in separate file format)

13.4 Mock Test Paper (Mini Assessment)

Instructions:
Write the HTML code for the following:

1. A basic page structure with title, heading, and footer
2. Create a contact form with text input, email input, and a submit button
3. Create a table showing 3 student names and their marks
4. Embed a YouTube video using <iframe>
5. Add a navigation bar with 4 links using semantic tags

13.6 Final Practice Checklist

Task	Completed? ✓
Build a full portfolio	
Design a resume page	
Create 2 forms with validation	
Solve 30 MCQs from mock test	
Complete mock HTML test	
Practice with DevTools	
Use ARIA and semantic structure	

Summary

- This unit ensures students are **interview-ready** and **certification-ready**
- Covers MCQs, short answers, and mini-projects
- Encourages real-world application and debugging fluency