HTML & HTML-5

By : LAKSHMIKANT DESHPANDE

Introduction to Web

1) Overview of Web And Web Application

2) Overview of Mobile and Mobile Applications

3) Web VS Mobile

4) Web Designer vs HTML Developer

Overview of Web and Web Applications

What is the Web?

- A system of interlinked hypertext documents accessed via the internet.
- Allows sharing of information through browsers.

What is a Web Application?

- A software application that runs on a web server, accessed through browsers (e.g., Gmail, Facebook).

Components of Web Applications:

- **Front-end:** UI/UX, HTML, CSS, JavaScript (what the user interacts with).
- **Back-end:** Server-side logic, databases, APIs (managing data and business logic).
- **Full-Stack:** Combines both front-end and back-end development.

Benefits:

- Accessible from any device with a browser.
- Easy to update and maintain.

Overview of Mobile and Mobile Applications

What is Mobile?

- Devices like smartphones, tablets that use wireless technologies to access services.

What is a Mobile Application?

- Software designed to run on mobile devices (e.g., Instagram, WhatsApp).

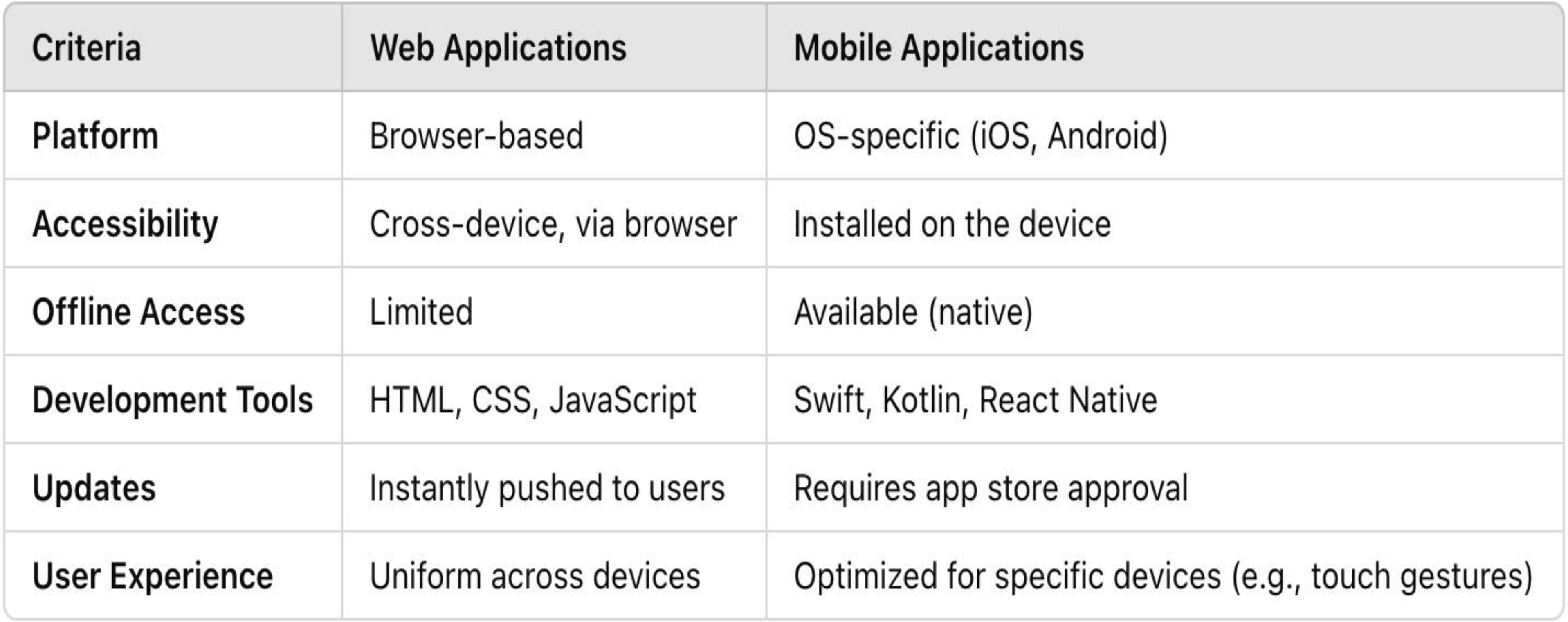
Types of Mobile Applications:

- **Native Apps:** Built for specific platforms (iOS/Android) using platform-specific languages (Swift, Kotlin).
- **Hybrid Apps:** Built using web technologies (HTML, CSS, JavaScript) but run in a webview within native apps (e.g., Ionic).
- **Progressive Web Apps (PWAs):** Web applications with mobile-like experience and capabilities.

Benefits:

- Rich user experience.
- Offline functionality for some apps.

Web vs Mobile



Criteria	Web Applications	Mobile Applications
Platform	Browser-based	OS-specific (iOS, Android)
Accessibility	Cross-device, via browser	Installed on the device
Offline Access	Limited	Available (native)
Development Tools	HTML, CSS, JavaScript	Swift, Kotlin, React Native
Updates	Instantly pushed to users	Requires app store approval
User Experience	Uniform across devices	Optimized for specific devices (e.g., touch gestures)

Web Designer vs HTML Developer

Web Designer:

- Focuses on **visual design** and **user experience**.
- Tools: Adobe XD, Figma, Sketch.
- Responsible for layout, typography, color schemes, and making the web app aesthetically pleasing.
- Works with wireframes, prototypes, and mockups.

HTML Developer:

- Focuses on **coding** the structure of the web page using **HTML**.
- Ensures the design translates to functional code.
- Works closely with CSS and JavaScript developers for implementation.
- Responsible for ensuring responsiveness, accessibility, and browser compatibility.

Key Difference:

- A web designer deals more with the creative aspect, while an HTML developer is responsible for the technical execution of that design.

Day - 2

Core Concepts of HTML

Definition of HTML

- HTML (HyperText Markup Language) is the standard language used to create and design webpages.
- It provides the structure and layout of a web page by using various elements such as headings, paragraphs, images, links, and more.
- Each element is represented by **tags** that help the browser understand how to display the content to the user.
    - **HyperText** refers to the way web pages are linked together using hyperlinks.
    - **Markup Language** means that HTML is not a programming language but rather a way to "mark up" text with tags to give it structure and meaning.

Elements and Attributes

HTML Elements are the building blocks of web pages. Every webpage is made up of various elements like headings (<h1>), paragraphs (<p>), images (<img>), links (<a>), etc. These elements help structure the content in a way that makes it readable and visually organized for the user.

Attributes provide additional information about HTML elements. They help define the characteristics and behavior of elements.

For example:

- The href attribute in a link (<a href="https://example.com">Click here</a>) defines the destination URL.
- The src attribute in an image tag (<img src="image.jpg">) defines the path to the image.
- The class and id attributes are used to style elements with CSS or target them with JavaScript.

Elements and Attributes contd…

Together, elements and attributes play a crucial role in:

- **Structuring content**: Organizing text, media, and interactive components for clarity.
- **Styling**: Attributes like class and id allow developers to apply styles efficiently.
- **User Experience**: Semantic elements improve accessibility, SEO, and overall user navigation.

HTML Tags

An HTML tag is a keyword enclosed in angle brackets (<>) used to define an HTML element on a webpage. Tags are essential in telling the browser how to display content, whether it's text, images, links, or other multimedia. Each tag defines a specific purpose and behavior for the element it wraps.

The syntax of an HTML tag:

- <tagname> content </tagname>

Basic HTML Document Structure

HTML documents consist of nested tags that create a hierarchy and structure for the content. Each document starts with a <!DOCTYPE> declaration and must include two key tags: <html> and <body>.

- **<!DOCTYPE html>**: Declares the document type as HTML5.
- **<html>**: The root element of the HTML document.
- **<head>**: Contains meta-information about the document (e.g., title, character set, links to stylesheets).
- **<body>**: Contains the visible content of the webpage.

Types of HTML Tags

1. Single Tags / Empty Tags / Void Tags / Self-Closing Tags

Single, empty, or self-closing tags are tags that do not need a closing tag because they do not enclose any content. These tags either stand alone or are self-contained, meaning they perform a specific function without requiring content inside them.

Examples:

- **<img>**: Defines an image to be displayed on the page.
- **<br>**: Inserts a line break.
- **<hr>**: Creates a horizontal rule or divider line.

Types of HTML Tags contd…

2. Paired Tags

Paired tags consist of an **opening tag** and a **closing tag**. Content is placed between the tags, and the closing tag is denoted by a forward slash (/).

Examples:

- **<h1></h1>**: The first-level heading.
- **<p></p>**: Defines a paragraph of text.

Types of HTML Tags contd…

3. Semantic Tags

Semantic tags provide meaning and context to both the browser and developers. They describe the purpose of the content enclosed within them, rather than just how it should appear.

Examples:

- **<header>**: Represents the introductory section of a webpage or article.
- **<article>**: Represents a self-contained composition, such as a blog post or news article.
- **<footer>**: Represents the footer section, often containing copyright information, contact links, etc.

Types of HTML Tags contd…

4. Non-Semantic Tags

Non-semantic tags do not describe the purpose of the content inside them. They are often used for layout and styling purposes but don’t carry any inherent meaning related to the content.

Examples:

- **<div>**: Used to group block-level content for styling or layout purposes.
- **<span>**: Used to group inline content for styling or scripting.

HTML Elements

An HTML element consists of a **start tag**, **content**, and an **end tag** (in most cases). Elements are the fundamental building blocks of HTML, defining the structure and meaning of web content. They may also include attributes that provide additional information or functionality.

Types of HTML Elements

1. Inline Elements

Display Characteristics:

- Inline elements do not start on a new line; they flow along with the text.
- They only take up as much width as their content requires.
- Inline elements typically hold small pieces of content, such as formatting text within a paragraph.

Examples of Inline Elements:

- **<span>**: A generic inline container, used for applying styles.
- **<a>**: Defines a hyperlink.
- **<strong>**: Makes text bold, giving it importance.
- **<em>**: Emphasizes text, usually rendering it in italics.

Types of HTML Elements contd…

2. Block-Level Elements

Display Characteristics:

- Block-level elements always start on a new line and take up the full width available (by default).
- They are used to create larger structures on the webpage, such as sections, headings, or paragraphs.

Examples of Block-Level Elements:

- **<div>**: A generic block container, often used to group elements for styling.
- **<p>**: Defines a paragraph.
- **<h1> to <h6>**: Heading tags, with <h1> being the highest level and <h6> the lowest.

Types of HTML Elements contd…

3. Inline-Block Elements

Display Characteristics:

- Inline-block elements are a combination of inline and block behaviors. They sit inline with the surrounding content, like inline elements, but they allow you to set width, height, and padding like block elements.
- They are useful when you want elements to sit side by side, yet maintain block-level properties.

Examples of Inline-Block Elements:

- **<button>**: Creates an interactive button.
- **<img>**: Embeds an image.

Summary of HTML Elements

- **Inline Elements**: Flow within the text and take up only the space needed (e.g., <span>, <a>).
- **Block-Level Elements**: Take up the full width and start on a new line (e.g., <div>, <p>).
- **Inline-Block Elements**: Combine the behavior of inline and block elements, allowing layout control without breaking the text flow (e.g., <button>, <img>).

Understanding these different types of elements and how they behave is crucial for structuring and designing webpages effectively. Each type has a specific use case, and using them correctly helps create well-organized and responsive layouts.

Introduction to HTML Attributes

HTML attributes provide additional information about elements, helping to modify their behavior, content, or appearance. Attributes are always added to the **opening tag** of an element and consist of a name-value pair.

Syntax:

```
<tag attribute="value"></tag>
```

- **attribute**: The name of the attribute.
- **value**: The value or data assigned to the attribute.

Types of Attributes

1. Predefined Attributes

Predefined attributes are standard attributes that are commonly used in HTML elements. They serve specific purposes such as defining links, image sources, and accessibility information.

Examples:

- **href**: Used with the <a> (anchor) tag to specify the URL of a link.
- **src**: Specifies the source file for images, scripts, or media (used in <img>, <script>, <iframe>, etc.).
- **alt**: Provides alternative text for an image, used for accessibility and search engines.

Types of Attributes contd…

2. Global Attributes

Global attributes are attributes that can be applied to any HTML element, regardless of its type. They are widely used to manage the style, behavior, or structure of an element.

Examples:

- **id**: Uniquely identifies an element on a page.
- **class**: Assigns one or more class names to an element, which can be used for styling via CSS or targeted by JavaScript.
- **style**: Directly applies inline CSS styles to an element.
- **data-***: Used to store custom data within an HTML element for access via JavaScript.

Types of Attributes contd…

3. Element-Specific Attributes

Definition: Element-specific attributes are attributes that are only relevant to particular elements. They control functionality or provide additional information specific to the element they modify.

Examples:

- **src**: Used for specifying the source of an image or a script (e.g., <img src="path/to/image.jpg">).
- **alt**: Provides alternative text for images for accessibility.
- **type**: Specifies the type of input field (e.g., <input type="text">, <input type="submit">).

HTML Attributes Summary

HTML Attributes modify elements and provide additional functionality.

Predefined Attributes: Attributes like href, src, and alt are used frequently to specify URLs, image sources, and alternative text.

Global Attributes: id, class, and style are widely used across all HTML elements for identifying, styling, and organizing content.

Element-Specific Attributes: Attributes like src and type apply to specific elements like <img> and <input>, giving those elements extra functionality.

Day - 3

HTML Tags

Common Text Tags in HTML

Headings Tags: Used to define headings and subheadings.

<h1> to <h6>: These represent headings, with <h1> being the highest (most important) and <h6> being the lowest (least important).

- <h1>Main Heading (h1)</h1>
- <h2>Subheading (h2)</h2>
- <h3>Section Subheading (h3)</h3>
- <h4>Details (h4)</h4>
- <h5>Minor Details (h5)</h5>
- <h6>Least Important Information (h6)</h6>

HTML Paragraph: <p>

- The <p> tag is used for defining a paragraph in HTML. It's the primary way to create a block of text in your webpage.

Lists in HTML

Lists help in organizing items, either in a specific order or without a specific order. HTML provides three types of lists:

1. Ordered List (<ol>)
   a. The <ol> tag creates an ordered list, and each list item is wrapped inside the <li> (list item) tag.
   b. type: Defines the type of numbering (1, A, a, **I**, i).
   c. start: Specifies the starting number or letter for the list.
2. Unordered List (<ul>)
   a. The <ul> tag creates an unordered list, and each item inside the list is enclosed within the <li> tag.
   b. type: You can change the bullet style using the type attribute (disc, circle, square).
3. Definition List (<dl>)
   a. A **definition list** is used for creating lists of terms and their definitions.
   b. **<dl>**: The container for the definition list.
   c. **<dt>**: Defines the term (or name) to be defined.
   d. **<dd>**: Provides the description (or definition) of the term.

Table

An HTML table is defined using the <table> element, with rows created using <tr> (table row) and individual cells using <td> (table data). For headers, the <th> (table header) element is used.

- <table>: The parent element that wraps all table content.
- <tr>: Defines a table row.
- <th>: Defines a table header, usually rendered as bold and centered text.
- <td>: Defines a cell of table data, where actual content resides.
- border="1": Adds borders to the table and its cells.
- <caption>: Adds a caption or title to the table, appearing at the top of the table

Table Attributes

- **border**: Adds a border to the table.
- **cellpadding**: Adds space inside the cells (between the cell content and the cell borders).
- **cellspacing**: Adds space between the table cells.
- **width**: Specifies the width of the table.
- **align**: Aligns the table horizontally (left, right, or center).
- **Column Span (colspan)**: a cell can span across multiple columns using the colspan attribute.
- **Row Span (rowspan)**:a cell can span across multiple rows using the rowspan attribute.

Table Sections

- **<thead>**: Table header section.
- **<tbody>**: Table body section.
- **<tfoot>**: Table footer section.

Article and Aside

The **<article>** tag in HTML is used to represent a self-contained piece of content that could stand alone or be independently distributed. An article could be a blog post, news article, forum post, or other types of content that have meaning on their own.

The **<aside>** tag represents content that is tangentially related to the main content. This content could be supplementary or related information, such as a sidebar, quotes, or advertisements.

Graphic Tags

<img> (Image) Tag

- The **<img>** tag in HTML is used to embed images in a webpage. It is a self-closing tag. Images enhance the visual appeal and usability of websites, and they can be used for various purposes such as branding, informative diagrams, or decorative visuals.
- **Attributes**: It uses attributes like src, alt, width, and height to control the display and description of the image.

<figure> Tag

- The **<figure>** tag is used to group images, illustrations, diagrams, or any media along with a caption. It helps associate an image with its caption, making it more semantic and accessible. The <figure> tag is typically used with the <figcaption> tag, which provides a caption or description of the content.
- **Semantic Grouping**: <figure> groups the image and its associated caption.
- **Supports Various Media**: It can include images, diagrams, videos, code snippets, etc.
- **Self-Contained Content**: <figure> content is independent of the main content but can support it.

Links

Anchor (<a>) Tag

The **<a>** tag, also known as the "anchor" tag, is used to create hyperlinks in HTML. These links can navigate to other pages, sections of the same page, or external websites. Links are a crucial part of web navigation and allow users to explore related content.

- **href="https://www.google.com"**: This links to Google’s website.
- **target="_blank"**: Opens the link in a new tab, so the user stays on the current page.
- Hash-Based Navigation (Within the Same Page)
- **download Attribute**:
- **Email Links**:
- **Telephone Links**:

Multimedia Tags in HTML

Common Multimedia Tags:

- **<audio>**: Used for embedding audio files.
- **<video>**: Used for embedding video files.
- **<embed>**: Used to embed external content like PDFs, flash content, or applications.
- **<object>**: Used to embed multimedia, such as images, videos, or other resources, and also to handle plugins.
- **<iframe>**: Used to embed another HTML page or interactive media like Google Maps or YouTube videos.

Multimedia Tags contd…

The **<audio>** tag is used to embed sound content, such as music tracks, podcasts, or any audio files.

- It supports multiple formats like **MP3**, **WAV**, and **OGG**.
- It allows playback control (play, pause, volume control).

Common Attributes:

- **controls**: Adds play, pause, and volume controls for the user.
- **autoplay**: Automatically plays the audio as soon as the page loads.
- **loop**: Loops the audio, restarting it when it finishes.
- **muted**: Mutes the audio on load.

Multimedia Tags contd…

The **<video>** tag is used to embed video content on a webpage. It supports a variety of formats and provides native controls for playing, pausing, and seeking.

- Supported formats: **MP4**, **WebM**, **OGG**.
- The tag allows the use of subtitles, captions, and multiple sources.

Common Attributes:

- **controls**: Adds controls like play, pause, and volume.
- **autoplay**: Automatically plays the video when the page loads.
- **loop**: Restarts the video when it ends.
- **muted**: Mutes the video sound.
- **poster**: Defines an image that will be displayed before the video starts playing.

Multimedia Tags contd…

The **<embed>** tag is used to embed external content like PDFs, Flash animations, or other multimedia files. It is a versatile tag that allows the embedding of various media types directly in a webpage.

- **src**: Specifies the URL of the external resource to be embedded.
- **width** and **height**: Define the size of the embedded resource.
- **type**: Specifies the media type of the embedded content (e.g., application/pdf, video/mp4).

The **<object>** tag is used to embed multimedia files such as images, videos, PDFs, or even Java applets. It can be more powerful than the <embed> tag because it allows fallback content to be displayed if the object fails to load.

- **data**: Specifies the URL of the resource to be embedded.
- **type**: Defines the type of the embedded content.
- **width** and **height**: Set the size of the embedded object.

Multimedia Tags contd…

The **<iframe>** (Inline Frame) tag is used to embed another HTML page or external content like YouTube videos, Google Maps, or other web pages within the current webpage.

- The embedded content can be a complete webpage or a part of an external website.
- **Sandboxing** can be applied to restrict the behavior of the embedded content for security reasons.

Common Attributes:

- **src**: The URL of the content to be embedded.
- **width** and **height**: Dimensions of the iframe.
- **frameborder**: Specifies whether or not the iframe should have a border.
- **allowfullscreen**: Allows the iframe to be viewed in fullscreen mode.

Layout Tags

The **<div>** tag is a generic container used to group elements for styling or scripting purposes. It does not have any semantic meaning on its own but is useful for applying styles or scripts to a group of elements.

- **Block-Level Element**: It occupies the full width available and starts on a new line.
- **Flexible**: Widely used in CSS for layout and design purposes.

The **<section>** tag is used to define a thematic grouping of content, typically with a heading. It indicates that the content inside is related and should be treated as a standalone section of the document.

- **Semantic Element**: Provides meaning about the content it contains.
- **Used for Grouping**: Suitable for creating sections within a webpage, such as chapters, topics, or distinct areas of information.

The **<header>** tag is used to define introductory content for a page or section. It typically contains headings, logos, navigation links, or introductory text.

- **Semantic Element**: Indicates the header of a document or section.
- **Multiple Headers**: You can have multiple <header> elements in a page.

Layout Tags

The **<main>** tag is used to specify the main content of a webpage. It should be unique to the document and contain the central content of the page.

- **Semantic Element**: Helps search engines and assistive technologies understand the primary content of the page.
- **Single Instance**: There should be only one <main> tag per document.

The **<nav>** tag is used to define a navigation section in a webpage. It typically contains links to other pages or sections within the same page.

- **Semantic Element**: Indicates that the enclosed links are for navigation.
- **Accessibility**: Helps users understand the navigation structure of the website.

The **<footer>** tag is used to define the footer of a document or section. It usually contains information about the author, copyright information, links to related documents, or other metadata.

- **Semantic Element**: Indicates the footer section of a document or section.
- **Multiple Footers**: You can have multiple <footer> elements in a page, such as one for the main content and another for specific sections.

Summary of Layout Tags

<div>: A generic container for grouping elements and applying styles.

<section>: A thematic grouping of content, typically with a heading.

<header>: Introductory content, usually containing headings and navigation.

<main>: The main content of the document, unique to the page.

<nav>: A navigation section for links to other pages or sections.

<footer>: The footer of a document or section, containing metadata or links.

Day - 4

Forms

Forms

Forms in HTML are sections of a webpage that allow users to input data. They are a crucial part of interactive web applications, enabling users to submit information to a server or perform specific actions. Forms can contain various input elements, such as text fields, radio buttons, checkboxes, dropdowns, and buttons.

Common Elements in Forms

- **Text Input**: For single-line text entries (e.g., name, email).
- **Textarea**: For multi-line text input (e.g., comments, messages).
- **Radio Buttons**: For selecting one option from a set.
- **Checkboxes**: For selecting multiple options.
- **Select Dropdown**: For selecting an option from a list.
- **Buttons**: For submitting the form or performing actions.

Why Should We Create Forms?

User Interaction: Forms allow users to interact with your website, providing a way for them to express their needs, preferences, and feedback.

Data Collection: Forms are a primary method for collecting data from users. Whether it’s user registrations, contact forms, surveys, or order placements, forms help gather crucial information for analysis and decision-making.

User Authentication: Forms are used for user sign-up and login processes, allowing users to create accounts and secure access to personalized content.

Feedback Mechanism: Forms provide a way for users to submit feedback, questions, or concerns, enabling businesses to improve their services and address user needs.

E-commerce Transactions: In online shopping, forms are necessary for capturing user information, payment details, and shipping addresses during the checkout process.

Search Functionality: Forms can facilitate search operations, enabling users to enter keywords or filters to find specific content on a website.

Enhanced User Experience: A well-designed form improves user experience by simplifying the process of submitting information and providing instant feedback.

Form Tags and Form Attributes

The **<form>** tag wraps all the input elements, labels, buttons, and other interactive elements in a form.

Form Attributes : Form attributes modify the behavior of the form and control how data is sent to the server.

- **method** : Specifies how to send form data to the server.
  - **GET**: Appends form data to the URL in name/value pairs. It's suitable for retrieving data (e.g., search queries).
  - **POST**: Sends form data in the request body, suitable for submitting sensitive information (e.g., passwords).
- **action** : Specifies the URL where the form data will be sent when the form is submitted.
- **enctype** : Specifies how the form data should be encoded when submitting it to the server. It is especially important when dealing with file uploads.
  - **application/x-www-form-urlencoded**: Default value, encodes the data as key/value pairs.
  - **multipart/form-data**: Used for forms that include file uploads.
  - **text/plain**: Sends data as plain text.

Form Attributes contd…

- **autocomplete** : Specifies whether the browser should enable autocomplete for the form fields.
  - **on**: Enables autocomplete (default).
  - **off**: Disables autocomplete.
- **novalidate** : This Boolean attribute prevents the browser from performing validation on the form before submission. It’s useful when you want to handle validation manually through JavaScript.
- **target** : Specifies where to display the response after submitting the form.
  - _self, _blank, _parent, _top
- **id** : Unique identifier for the form, used for linking and styling.
- **name** : Name of the form, used to reference the form in scripts.
- **class** : Specifies one or more class names for styling the form with CSS.
- **style** : Inline CSS styles for the form.
- **onsubmit** : JavaScript event handler that executes when the form is submitted.
- **onreset** : JavaScript event handler that executes when the form is reset.
- **accept-charset** : Specifies the character encodings that the server accepts for the form data.

Form Elements & Form Element Attributes

- **What are HTML Form Elements?**
  - Interactive elements that allow users to input data.
  - Used for gathering information from users (e.g., contact forms, surveys).
- **Importance**:
  - Essential for web applications.
  - Facilitate user interaction with websites.
- **Attributes**:
  - **action**: URL where form data is sent.
  - **method**: HTTP method used to send data (GET or POST).
  - **enctype**: Encoding type for form data.
  - **target**: Where to display the response.

Elements used in forms

- The <input> Element : Accepts various types of user input.

Common Types:

- text, password, checkbox, radio, email, number, file.

Attributes:

- **name**: Name for the input field.
- **type :** Specifies the input type (e.g., text, password, checkbox).
- **value**: Default value for the input.
- **placeholder**: Hint for the user.
- **required**: Indicates mandatory fields.

Input Types

- **Text**: Single-line text input.
- **Password**: Hides input characters for sensitive data.
- **File**: Allows users to upload files.
- **Hidden**: Stores data not visible to the user.
- **Radio**: Lets users select one option from a group.
- **Checkbox**: Allows multiple selections from a set of options.
- **Submit (Action Item)**: Submits the form data to a specified action.
- **Reset (Action Item)**: Resets form fields to their default values.

Elements used in forms contd…

The <textarea> Element : Allows multi-line text input.

Attributes:

- **name**: Name of the textarea.
- **rows**: Number of visible rows.
- **cols**: Number of visible columns.
- **placeholder**: Hint for the user.

Elements used in forms contd…

Dropdown Box

The <select> Element : Dropdown list for selecting options.

Attributes:

- **name**: Name for the select field.
- **multiple**: Allows selection of multiple options.
- **required**: Indicates an option must be selected.

The <option> Element : Defines an option within a <select>.

Attributes:

- **value**: Value submitted when the option is selected.
- **selected**: Default selected option.
- **disabled**: Prevents selection.

Day - 5

HTML 5

HTML 5

- HTML5 brought significant improvements over older versions of HTML (HTML 4.01 and XHTML).
- It simplified syntax, introduced semantic elements like <header>, <footer>, and <article>, and enhanced multimedia handling with native support for <audio> and <video>.
- Forms were upgraded with new input types like email, url, and built-in validation.
- HTML5 also introduced graphics support with <canvas>, offline capabilities with local storage and service workers, and new APIs for real-time web interactions.
- Deprecated presentational elements were removed in favor of CSS, making HTML5 more focused on structure and meaning.
- Additionally, it offered better support for mobile and responsive design, making it suitable for modern web development.

Feature	HTML 4.01/XHTML	HTML5
Doctype Declaration	Long and complex (<!DOCTYPE HTML PUBLIC...>)	Simple (<!DOCTYPE html>)
Semantic Elements	Lacked dedicated tags for structure (relied on <div>)	Introduced <header>, <footer>, <article>, etc.
Multimedia Support	Required third-party plugins (Flash)	Native <audio>, <video> elements
Form Input Types	Limited (text, password, submit)	New types: email, url, number, date, etc.
Form Validation	Required JavaScript for validation	Built-in validation with required, pattern
Graphics Support	No built-in support	<canvas> element for 2D graphics
Local Storage	Cookies for storage	localStorage and sessionStorage APIs
APIs	Minimal API support	New APIs: Geolocation, Drag-and-Drop, WebSocket
Deprecated Elements	Used tags like <center>, <font>, <big>	These elements are removed, favoring CSS for styling
Offline Support	No built-in offline features	AppCache, Service Workers for offline capabilities
Character Encoding	Complex syntax (<meta http-equiv=...>)	Simple encoding (<meta charset='UTF-8'>)
Mobile and Responsive Design	Lacked mobile-first design features	<meta name='viewport'> for responsive design
Error Handling	Strict, hard to recover from syntax errors	Lenient error handling, renders even with errors
Browser Compatibility	Inconsistent rendering across browsers	Improved cross-browser compatibility

HTML5 Features at a Glance

- **Simple DOCTYPE Declaration**: Easily declare your HTML5 document with <!DOCTYPE html>.
- **Character Encoding**: Use <meta charset="UTF-8"> for simple character encoding.
- **Multimedia Support**: Embed audio with <audio>, video with <video>, and create graphics using <svg> and <canvas>.
- **Client-side Data Storage**: Store user data easily with localStorage.
- **Interactive Documents**: Enhanced forms and user interactivity for better engagement.

HTML5 Features Overview

- **New Structural Elements**: Use semantic tags like <article>, <header>, <footer>, <nav>, <section>, and <figure> for better document structure and accessibility.
- **Enhanced Form Controls**: Implement new input types such as calendar, date, time, email, URL, and search for improved user experience.
- **JavaScript Enhancements**: Leverage improved JavaScript integration for interactive and dynamic web applications.
- **New HTML5 APIs**: Access powerful new APIs to enhance functionality, including multimedia handling and enhanced user interactions.
- **HTML Geolocation**: Utilize the Geolocation API to retrieve the user's location and provide location-based services.
- **HTML Drag and Drop**: Implement drag-and-drop functionality for a more interactive and user-friendly interface.
- **Web Storage**: Store data client-side with localStorage and sessionStorage for persistent and temporary data storage.

HTML5 Validation Attributes

- **required**: Ensures that a user fills out the field before submitting the form.
- **pattern**: Specifies a regular expression that the input's value must match for valid submission.
- **min**: Sets the minimum value for input types like number, date, and range.
- **max**: Defines the maximum value for inputs such as number, date, and range.
- **minlength**: Specifies the minimum number of characters required in a text input.
- **maxlength**: Limits the maximum number of characters a user can enter in a text input.
- **type**: Defines the type of input (e.g., email, url) which automatically applies built-in validation.
- **step**: Determines the legal number intervals for numeric input, such as allowing values like 0.5 in a range.
- **novalidate**: Disables the browser's default validation for the form, allowing for custom validation handling.

New Structural Elements

HTML5 introduced new semantic elements that improve the readability of code and make it easier for search engines and screen readers to understand the structure of a web page.

- <header>: Represents the header of a section or page.
- <footer>: Represents the footer of a section or page.
- <nav>: Defines navigation links.
- <article>: Represents independent, self-contained content.
- <section>: Groups related content together, similar to <div>, but with semantic meaning.
- <aside>: Represents content that is tangentially related to the main content (like a sidebar).
- <figure>: Encapsulates media (images, diagrams, etc.) and captions.
- <figcaption>: Provides a caption for <figure> content.
- <main>: Represents the main content of the page

New Form Controls

HTML5 introduced new input types and attributes that simplify form handling and validation.

- **New Input Types**:
  - type="email": For entering email addresses.
  - type="url": For entering web addresses.
  - type="date": For selecting dates.
  - type="time": For selecting times.
  - type="datetime-local": For selecting date and time (without time zone).
  - type="range": For selecting a numeric value within a range (slider).
  - type="number": For entering numeric values.
  - type="tel": For entering phone numbers.
  - type="search": For search inputs.
  - type="color": For picking colors.

New Form Controls contd…

Form Attributes:

- autocomplete: Suggests previously entered values.
- novalidate: Disables form validation.
- required: Specifies that an input field must be filled out before submitting.
- placeholder: Provides a hint to the user about what to enter in the field.
- pattern: Specifies a regular expression pattern that the input must match.

Multimedia Elements

HTML5 introduced native support for embedding media without the need for external plugins like Flash.

- <audio>: Embeds audio files directly into a web page.
    - Supports multiple formats like MP3, OGG, and WAV.
    - Attributes like controls, autoplay, and loop.
- <video>: Embeds video files directly into a web page.
    - Supports formats like MP4, WebM, and OGG.
    - Attributes like controls, autoplay, loop, and poster.
- <source>: Used with <audio> and <video> to define multiple sources for the media.
- <track>: Provides text tracks (like subtitles or captions) for <video> and <audio> elements.

Graphics and Interactive Content

HTML5 introduced new ways to display graphics and interactive content natively in the browser.

- **Scalable Vector Graphics (SVG)**: A language for describing 2D graphics and graphical applications in XML.
  - <svg>: Embeds SVG content for vector-based graphics.
- **Canvas API**: Provides a way to draw 2D shapes and bitmap images dynamically using JavaScript.
  - <canvas>: Defines a drawing area for graphics, charts, games, etc.

New APIs (Application Programming Interfaces)

HTML5 introduced several APIs that allow developers to interact with the browser and user data in powerful ways.

- **Geolocation API**: Allows web applications to obtain the geographical location of the user.
  - Methods like getCurrentPosition() and watchPosition().
- **Web Storage API**:
  - **Local Storage**: Stores data persistently in the user's browser.
  - **Session Storage**: Stores data temporarily (for the session duration).
  - Replaces traditional cookies for client-side data storage.
- **Web Workers**: Allows background threads to run JavaScript code without affecting the performance of the main page.

New APIs (Application Programming Interfaces) contd…

- **Drag and Drop API**: Enables drag-and-drop functionality within the browser.
- **History API**: Allows manipulation of the browser session history using pushState(), replaceState(), and popState().
- **WebSockets API**: Provides full-duplex communication between the browser and server for real-time applications.
- **Server-Sent Events (SSE)**: Enables a server to send updates to the client in real time using the <eventsource> element.

Client-Side Data Storage

HTML5 offers new methods for storing data on the client-side:

- **Local Storage**: Allows for storing large amounts of data (strings) with no expiration date.
- **Session Storage**: Similar to local storage but data is deleted when the session ends (when the page or tab is closed).
- **IndexedDB**: A low-level API for storing significant amounts of structured data (in key-value pairs) on the client.

Offline and Cache Features

HTML5 provides functionality that allows web applications to work offline:

- **Application Cache**: Allows a web application to be cached and accessed even without an internet connection.
  - Deprecated in favor of **Service Workers** in modern web development.
- **Service Workers** (introduced after HTML5): Scripts that run in the background to provide features like offline functionality, background sync, and push notifications.

HTML5 Validation Attributes

HTML5 simplifies form validation with built-in validation attributes:

- **Required Fields**: required attribute ensures the field must be filled out before form submission.
- **Pattern Matching**: pattern attribute ensures the input value matches a specified regular expression.
- **Min/Max Length**: minlength and maxlength attributes define limits on input field length.
- **Step**: Defines intervals for numerical inputs (type="number", type="range").

Microdata and Custom Data Attributes

HTML5 introduced new ways to add metadata to HTML elements:

- **Microdata**: Provides a standardized way to describe content through additional tags and attributes to improve SEO and data parsing (e.g., with search engines).
- **Custom Data Attributes**: Attributes prefixed with data- to store private data that can be accessed via JavaScript (data-* attributes).
  - Example: <div data-user-id="12345"></div>.

Interactive Documents

HTML5 enhances the interactivity and dynamic nature of web documents:

- **Details and Summary Elements**: Allows content to be hidden or revealed on demand.
  - <details>: Encapsulates details the user can view or hide.
  - <summary>: Defines a visible heading for the <details> element.
- **Meter and Progress Elements**:
  - <meter>: Represents a scalar measurement within a known range (e.g., a disk space gauge).
  - <progress>: Represents the completion progress of a task (e.g., a progress bar).

Native Browser Capabilities

HTML5 exposes several new capabilities directly through the browser:

- **Content Editable**: The contenteditable attribute allows an element’s text to be edited by the user.
- **Autofocus**: Automatically sets focus on an input field when a page loads.
- **Autoplay**: Starts media playback (video or audio) as soon as the page is loaded (though modern browsers limit its use due to user experience concerns).
- **Placeholder**: Displays a hint in an input field before the user enters data.

Accessibility Enhancements

HTML5 focuses on accessibility and making web applications more inclusive for users with disabilities:

- **ARIA Roles**: HTML5 enhances accessibility via better support for ARIA (Accessible Rich Internet Applications) attributes, helping screen readers and other assistive technologies understand and navigate the content.