Introduction to HTML and Its Structure

-> HTML (HyperText Markup Language) is the standard language used to create and design webpages.

-> It defines the structure of web content through a system of tags and elements. Each tag in HTML tells the browser how to display the content enclosed within it.

A basic HTML document has the following structure:

```
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
</head>
<body>
  <h1>This is a Heading</h1>
  <p>This is a paragraph of text.</p>
</body>
</html>
```

Explanation of Structure:

1 **<!DOCTYPE html>** – Declares the document type as HTML5.

2 **<html>** – The root element that contains the entire HTML document.

3 **<head>** – Contains meta-information about the webpage (title, styles, etc.).

4 **<title> –** Sets the title of the webpage (appears on the browser tab).

5 **<body>** – Contains all the visible content of the webpage.

Explanation of Key HTML Tags.

1. Anchor Tag (<a>)

Used to create hyperlinks that link one page to another.

Example:

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

Attributes:

href: Specifies the URL of the page the link goes to.

target: Defines where to open the link (_blank opens it in a new tab).

2. Form Tag (<form>)

The <form> tag is used to collect user input and send it to a server for processing.

Example:

```
<form action="/submit" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="username">
  <input type="submit" value="Submit">
</form>
```

Attributes:

action: URL where form data is sent.

method: Defines how data is sent (get or post).

3.Table Tag (<table>)

The <table> tag is used to display data in rows and columns.

Example:

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>John</td>
    <td>25</td>
  </tr>
</table>
```

Attributes:

<tr>: Table row.

<th>: Table header.

<td>: Table data cell.

4.Image Tag (<img>)

The <img> tag is used to display images on a webpage.

Example:

```
<img src="image.jpg" alt="Description" width="300" height="200">
```

Attributes:

src: Specifies the image source (path or URL).

alt: Alternative text displayed if the image fails to load.

width/height: Defines image dimensions.

5. List Tags (<ul>, <ol>, <li>)

Used to display items in a structured list.

Unordered List Example:

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

Ordered List Example:

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

6.Paragraph Tag (<p>):

Defines a paragraph of text.

Example:

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

**7.Line Break (
)**

Inserts a single line break. It does not require a closing tag.

Example:

```
<p>This is the first line.<br>This is the second line.</p>
```

8.Label Tag (<label>)

Defines a caption for an input field.

Example:

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

</form>

Overview of CSS and Its Importance in Web Design

Cascading Style Sheets (CSS) is a stylesheet language used to control the presentation and layout of HTML elements on a web page. While HTML structures the content, CSS determines how that content looks — including colors, fonts, spacing, and positioning.

Importance of CSS in Web Design:

1. CSS separates content (HTML) from design, making web pages easier to maintain.

2. It enables consistency across multiple web pages by using shared style rules.

3. CSS improves website performance by reducing code redundancy.

4. It allows responsive web design, ensuring compatibility across different devices and screen sizes.

5. Designers can easily change the entire look of a website by modifying a single CSS file.

Types of CSS

1. Inline CSS

Inline CSS is applied directly to individual HTML elements using the style attribute. It is useful for quick styling of specific elements but not recommended for large-scale projects as it mixes content with design.

Example: This is a blue paragraph.

2. Internal CSS

Internal CSS is defined within a <style> tag inside the <head> section of an HTML document. It is suitable for styling a single page without affecting others.

Example:
```
<html>
<head>
<style>
 p { color: green; font-size: 20px; }
</style>
</head>
 <body>
<p>This is a green paragraph.</p>
```

</body>
</html>

3. External CSS

External CSS is written in a separate file with a .css extension and linked to the HTML document using the <link> tag. It is the most efficient and preferred method for large websites as it ensures consistency and reusability.

Example:

HTML File:

```
<html>
<head>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<p>This paragraph is styled using external CSS.</p>
</body>
</html>
```

CSS File (styles.css):

```
p {
color: red; font-size: 22px;
}
```

Definition and Difference Between Margin and Padding.

-> In CSS, margin and padding are spacing properties used to control the layout and appearance of elements.

-> Although both are used to create space, they serve different purposes in how space is applied around or inside an element.

Definition:

1 Margin: The margin is the space outside an element's border. It creates distance between the element and other elements surrounding it.

2 Padding: The padding is the space inside an element's border. It creates space between the content and the element's border.

Difference Between Margin and Padding:

1. Margin controls the outer spacing of an element, while padding controls the inner spacing.
2. Changing the margin affects the space between different elements; changing the padding affects the space within the same element.
3. Margins can have negative values to overlap elements, while padding values cannot be negative.
4. Padding affects the background color and border area; margin does not.

Example:

```
<style>
div {
  border: 2px solid blue;
  margin: 20px;
  padding: 10px;
}
</style>
<div>This box has a margin outside and padding inside.</div>
```

Explanation:

The margin: 20px; creates 20 pixels of space outside the blue border, separating the element from other elements.

The padding: 10px; creates 10 pixels of space inside the border, pushing the content inward from the border.

Introduction to CSS Pseudo-Classes

CSS pseudo-classes are special selectors that define the state of an HTML element. They allow developers to apply styles based on user interaction or element state, such as when a user hovers over, focuses on, or clicks an element. Pseudo-classes begin with a colon (:) followed by the pseudo-class name.

Common CSS Pseudo-Classes

:hover

The :hover pseudo-class applies when a user moves the mouse pointer over an element.

Example:

```
a:hover {
  color: red;
  text-decoration: underline;
}
```

Explanation:
When the user hovers over a link (<a> tag), its color changes to red, and an underline appears.

:focus

The :focus pseudo-class applies when an element (such as a text input) is active or selected for input.

Example:

```
input:focus {
  border-color: blue;
  background-color: lightyellow;
}
```

Explanation:
When the input field is active or clicked, it gets a blue border and a light yellow background.

:active

The :active pseudo-class applies when an element is being clicked or activated.

Example:

```
button:active {
  background-color: darkgreen;
  color: white;
}
```

Explanation:
When the button is pressed (before releasing the mouse), it changes its background to dark green and text color to white.

:visited

The :visited pseudo-class applies to links that the user has already clicked.

Example:

```
a:visited {
  color: purple;
}
```

Explanation:
After a link has been visited, it appears in purple to indicate it's already been accessed.

:first-child / :last-child

These pseudo-classes target the first or last child element within a parent.

Example:

```
li:first-child {
  color: green;
}
li:last-child {
  color: blue;
}
```

Explanation:
The first list item will be green, and the last one blue.

Using Pseudo-Classes to Style Elements Based on Their State

Pseudo-classes enhance interactivity and visual feedback for users. They make web interfaces dynamic without needing JavaScript.

Example:

```
button {
   background-color: lightgray;
   border: none;
   padding: 10px 15px;
}
button:hover {
   background-color: lightblue;
}
button:active {
   background-color: blue;
   color: white;
}
```

Explanation:

-When the button is in its normal state, it's light gray.

-When hovered, it turns light blue.

-When clicked, it becomes blue with white text — giving users feedback that their action was recognized.

Difference Between ID and Class in CSS

In CSS, both id and class selectors are used to apply styles to HTML elements. However, they differ in how they are used and how many times they can be applied within a document.

Definition:

- The id selector is used to apply styles to a single unique element on a page. It is defined using the # symbol in CSS.

- The class selector is used to apply styles to multiple elements. It is defined using the . (dot)

symbol in CSS.

Example of ID Selector:

HTML:

```
<p id="uniqueText">This is a unique styled paragraph.</p>
```

CSS:

```
#uniqueText {
   color: blue;
   font-size: 18px;
}
```

Example of Class Selector:

HTML:

```
<p class="highlight">This paragraph is highlighted.</p>
<p class="highlight">This is another highlighted paragraph.</p>
```

CSS:

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

Key Differences:

Uniqueness: ID is unique within a page; a class can be used multiple times.

Selector Symbol: ID uses #, class uses . (dot).

Specificity: ID has higher specificity than class, meaning it overrides class styles if both are applied to the same element.

Use Case: ID is suitable for one-time elements like headers or main sections; class is ideal for styling multiple similar elements.

Usage Scenarios:

-> Use ID when you need to style or identify one specific element, such as a unique navigation bar or footer section.

-> Use Class when multiple elements share the same style, such as buttons, highlights, or cards.

Overview of Client-Server Architecture

The client-server architecture is a fundamental model in computer networking and web applications where multiple clients (users or devices) interact with a central server to access shared resources, data, or services. In this model, the client makes a request, and the server processes the request and sends back a response.

How It Works:

1. The client (e.g., a web browser or mobile app) sends a request to the server for data or

resources.

2. The server (e.g., a web or database server) receives the request, processes it, and generates a

response.

3. The response (e.g., a web page, image, or data) is sent back to the client through the network.

Example:

When you open a website:

- The client (your browser) sends a request for a webpage.
- The server (where the website is hosted) processes the request and sends back HTML, CSS, and JS files.
- The browser displays the website using the received data.

Difference Between Client-Side and Server-Side Processing.

Feature	Client-Side Processing	Server-Side Processing
Definition	Operations performed on the user's device (browser).	Operations performed on the web server before sending data to the client.
Languages Used	HTML, CSS, JavaScript	PHP, Python, Node.js, Java, ASP.NET
Execution Environment	Runs on the user's browser.	Runs on the server.
Performance	Reduces server load but depends on the user's device.	Handles heavy computation, ensuring data security and accuracy.
Security	Less secure; code is visible to users.	More secure; code and logic stay on the server.
Example	Form validation using JavaScript.	Fetching user data from a database.

Roles of a Client, Server, and Communication.

1. Client:

- Initiates a request for data or service.

- Examples: Web browsers, mobile apps, desktop applications.

- Converts server responses (HTML, JSON, XML) into user-friendly displays.

2. Server:

- Receives and processes client requests.

- Stores, manages, and delivers resources like webpages, data, and files.

- Examples: Web servers (Apache, Nginx), database servers (MySQL, MongoDB), application servers.

3. Communication Protocols:

- Define how clients and servers exchange information.

- Common protocols include:

• HTTP/HTTPS: Used for web communication.

• FTP: For file transfer.

• SMTP/IMAP/POP3: For email transmission.

• TCP/IP: Ensures reliable data transfer.

Introduction to the HTTP Protocol and Its Role in Web Communication.

The Hyper Text Transfer Protocol (HTTP) is the foundation of data communication on the World Wide Web. It defines how messages are formatted and transmitted between a client (such as a browser) and a server (where websites and applications are hosted).

When a user enters a website URL in their browser, the browser sends an HTTP request to the server. The server then processes this request and returns an HTTP response, which contains the requested data such as an HTML page, image, or JSON data.

Role of HTTP in Web Communication

1. Communication between client and server: HTTP enables communication between browsers, APIs, and web servers using a standardized request–response model.

2. Stateless protocol: Each HTTP request is independent: the server does not retain information from previous requests (though this can be managed using cookies or sessions).

3. Supports multiple methods: HTTP supports several methods, each serving a specific purpose:

- GET: Retrieve data from the server.

- POST: Send data to the server (e.g., form submission).

- PUT: Update existing data.

- DELETE: Remove data from the server.

- HEAD: Retrieve header information only.

4. Foundation for HTTPS: HTTPS (HTTP Secure) adds a layer of encryption using SSL/TLS to ensure secure data transmission.

HTTP communication consists of two parts:

1. HTTP Request — Sent by the client to the server.
2. HTTP Response — Sent by the server back to the client

Introduction to Servlets and How They Work.

-> A Servlet is a Java-based server-side program that extends the functionality of a web server.

-> It is used to create dynamic web content by processing requests and generating responses. Servlets are an essential component of Java EE for building web applications.

How Servlets Work

Servlets follow a specific lifecycle defined by the servlet container (like Apache Tomcat).

Here's how the process works:

Step	Description
1. Loading and Initialization	The web container loads the servlet class and calls its init() method. This happens once when the servlet is first requested.
2. Request Handling	Each client request triggers the service() method, which determines whether to call doGet(), doPost(), or another method based on the HTTP request type.
3. Response Generation	The servlet processes the request, interacts with databases or other components, and generates a dynamic response (often HTML or JSON).
4. Destruction	When the server shuts down or the servlet is no longer needed, the destroy() method is called to release resources.

Advantages of Servlets

1. **Platform Independence** — Written in Java, servlets run on any server with a JVM.

2. **Performance**— Servlets are faster than traditional CGI because they use threads instead of creating new processes for each request.

3. **Reusability and Modularity**— Components can be reused and easily maintained.

4. **Integration with Java Technologies** — Works seamlessly with JDBC, JSP, and EJB for enterprise-level applications.

5. **Security** — Provides robust security via HTTPS, authentication, and Java's security APIs.

Disadvantages of Servlets

1. **Complexity** — Writing servlets directly for large applications can be tedious compared to using frameworks like Spring or Struts.

2. **HTML Mixing** — Business logic and HTML generation may become intertwined, making code harder to maintain.

3. **Limited Presentation Capabilities** — Unlike JSP or modern front-end frameworks, servlets are not designed for UI design.

4. Requires Servlet Container— Needs a server like Tomcat, which adds setup and maintenance overhead.

Explanation of RequestDispatcher and the forward() and include() Methods .

The Request Dispatcher interface in Java Servlets allows a request to be forwarded from one resource (servlet, JSP, or HTML file) to another resource within the same web application. It promotes modular development by dividing web logic into smaller reusable components.

Methods of Request Dispatcher

The Request Dispatcher interface provides two primary methods:

Method	Description
Forward (request, response)	Transfers control to another resource on the server. The original servlet cannot
Include (request, response)	Includes the content of another resource (servlet, JSP, HTML) in the response.