# Unit-I: Introduction to Web and HTML

## Web Browsers

- **Definition**: Web browsers ek software application hote hain jo internet par websites ko access aur display karte hain. Yeh user ke liye ek interface provide karte hain jahan woh web pages ko dekh sake aur interact kar sake.
- **Kaise kaam karta hai?**
  1. User ek URL (Uniform Resource Locator) browser ke address bar me dalta hai. **Example:** www.google.com.
  2. Browser HTTP/HTTPS protocol ka use karke request ko server tak bhejta hai.
  3. Server response me HTML, CSS, aur JavaScript files bhejta hai.
  4. Browser in files ko render karke web page ko display karta hai.
- **Common Features**:
  - **Tabs:** Ek hi browser me multiple websites open karne ki facility.
  - **Bookmarks:** Favorite websites save karne ka option.
  - **Incognito Mode:** Private browsing feature.
  - **Developer Tools:** Web page debugging aur testing ke liye.
- **Examples**:
  - Google Chrome
  - Mozilla Firefox
  - Safari
  - Microsoft Edge
  - Opera

---

## Web Servers

- **Definition**: Web servers ek tarah ka software ya hardware system hote hain jo web browser se aane wali requests ko process karte hain aur data provide karte hain. Yeh internet ke peeche ka backend system hota hai.
- **Kaise kaam karta hai?**
  1. Jab ek browser request bhejta hai (via URL), toh server isse process karta hai.
  2. Agar requested content static hai (HTML files), toh server directly response bhejta hai.
  3. Agar dynamic content ki zarurat hai (jaise user ke input par depend karta hai), toh server scripting language (PHP, JSP) ka use karke dynamic response generate karta hai.
- **Functions**:
  - HTTP/HTTPS requests ko handle karna.
  - Static files jaise HTML, CSS, aur images serve karna.
  - Dynamic web pages create karna using server-side languages.
  - Logs maintain karna for security aur analytics.
- **Examples of Web Servers**:
  - **Apache**: Most widely used open-source web server.
  - **Nginx**: High-performance server for static content.
  - **Microsoft IIS (Internet Information Services)**: Windows-based server.
  - **LiteSpeed**: Fast and efficient server.

## Web Essentials

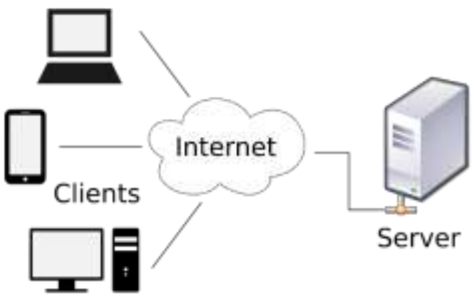
- Web essentials wo basic components hote hain jo internet aur websites ke kaam karne ke liye zaruri hote hain.

### (a) W3C (World Wide Web Consortium)

- Ek international organization hai jo web ke liye standards banata hai.
- **Purpose**:
  - Ensure karna ki websites har browser aur device par sahi kaam karein.
  - Standards jaise HTML, CSS, aur XML ko define karna.
  - Web accessibility ko promote karna.

### (b) Clients-Servers Architecture

- **Client**:
  - Wo device ya application hai jo server se data request karta hai.
  - **Example:** Web browser (Chrome, Firefox).
- **Server**:
  - Wo system ya software hai jo client ke requests process karta hai aur response bhejta hai.
  - **Example:** Apache, Nginx web servers.



### (c) HTTP and HTTPS

- **HTTP (HyperText Transfer Protocol)**:
  - Data transfer ka basic protocol.
  - **Example:** http://example.com.
- **HTTPS (HTTP Secure)**:
  - HTTP ke saath security layer (SSL/TLS) add karta hai.
  - **Example:** https://example.com.

### (d) DNS (Domain Name System)

- Domain names (jaise www.google.com) ko IP address me convert karta hai.
- **Example:** www.google.com → 142.250.190.78.

### (e) IP Address

- Internet Protocol Address wo unique identifier hota hai jo kisi device ya server ko assign hota hai.
- **Example:** IPv4: 192.168.1.1.

---

## Markup Languages – HTML

- **Definition**:
    - HTML ka full form hai **HyperText Markup Language**.
    - Yeh ek markup language hai jo web pages ka structure define karne ke liye use hoti hai.
    - HTML ka use karke hum text, images, videos, links, aur other elements ko web pages me add karte hain.
- **HyperText**:
    - Web pages me ek document se doosre document par jump karne ke liye links ka use hota hai.
    - **Example:** <a href="https://example.com">Click Here</a>
- **Markup Language**:
    - Tags ka use karke content ko structure aur formatting provide karte hain.
    - **Example:** <h1>Heading 1</h1>

---

## Tags

- **Definition**:
    - Tags HTML ka basic building block hote hain.
    - Ek tag start hota hai opening bracket < ke saath aur end hota hai closing bracket > ke saath.
- **Types of Tags**:
    1. **Paired Tags**:
        - Opening aur closing dono hote hain.
        - Example: <p>Paragraph</p>
    2. **Self-Closing Tags**:
        - Sirf ek hi tag hota hai.
        - Example: <img src="image.jpg" alt="Image">
- **Common Tags**:
    - <html>: Web page ka root element.
    - <head>: Metadata (title, CSS, scripts) define karta hai.
    - <body>: Web page ka visible content.
    - <h1> to <h6>: Headings define karne ke liye.
    - <p>: Paragraph ke liye.
    - <a>: Hyperlink ke liye.
    - <img>: Images ke liye.

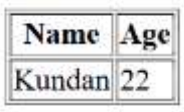
---

## Tables

- **Definition**:
  - Tables ka use structured data display karne ke liye hota hai (rows aur columns me).
  - Example: Timetables, Data charts.
- **Tags for Tables**:
  - <table>: Table ka container.
  - <tr>: Table row.
  - <th>: Table header.
  - <td>: Table data cell.
- **Example**:

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Kundan</td>
    <td>22</td>
  </tr>
</table>
```

**Output**



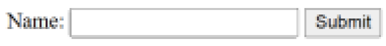
| Name | Age |
|---|---|
| Kundan | 22 |

## Forms

- **Definition**:
  - Forms user se input data collect karne ke liye use hote hain.
  - Example: Login form, Registration form.
- **Tags for Forms**:
  - <form>: Form ka container.
  - <input>: Data input karne ke liye (text, password, email, etc.).
  - <textarea>: Multi-line text input ke liye.
  - <button>: Form submit karne ke liye.
- **Example**:

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

**Output:**



## Frames

- **Definition**:
  - Frames web pages ko multiple sections me divide karte hain. Har section me alag-alag HTML content display ho sakta hai.
- **Tag for Frames**:
  - `<iframe>`: Ek page ke andar doosra web page embed karne ke liye.
- **Example**:

```
<iframe src="https://example.com" width="600" height="400"></iframe>
```

- **Note:** Frames ka use aaj outdated hai aur responsive designs me iframe preferred hota hai.

## XHTML

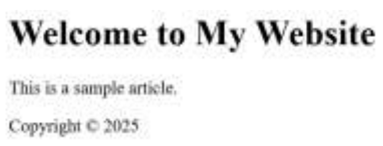
- **Definition**:
  - XHTML ka full form hai **Extensible HyperText Markup Language**.
  - HTML ka stricter version hai jo XML rules follow karta hai.
- **Features**:
  - Tags properly nested hone chahiye.
  - All tags lower case me likhe jaane chahiye.
  - Self-closing tags mandatory hote hain.
  - Example: `<br />`

## HTML5

- **Definition**:
  - HTML5 latest version hai jo advanced features aur better browser compatibility ke liye design kiya gaya hai.
- **Key Features**:
  - Multimedia Support: `<audio>` aur `<video>` tags.
  - Semantic Tags: `<header>`, `<footer>`, `<article>`, `<section>`.
  - Canvas API: Graphics aur animations ke liye `<canvas>`.
  - Geolocation API: User ki location detect karna.
- **Example of Semantic Tags**:

```
<header>
  <h1>Welcome to My Website</h1>
</header>
<article>
  <p>This is a sample article.</p>
</article>
<footer>
  <p>Copyright © 2025</p>
</footer>
```

**Output:**



```
Welcome to My Website

This is a sample article.

Copyright © 2025
```

# Unit-II CSS And Java Script

## CSS (Cascading Style Sheets)

- **Definition**:
  CSS ka use HTML elements ke appearance aur layout ko style karne ke liye hota hai. Yeh web pages ko visually appealing banata hai.
- **Why Use CSS?**
  - Web pages ko attractive aur user-friendly banata hai.
  - HTML structure aur styling ko alag karta hai.
  - Ek hi CSS file multiple web pages me use ki ja sakti hai.
- **Syntax of CSS**:
  CSS ka syntax selector aur declaration se banta hai:

```
selector {
  property: value;
}
```

  **Example**:

```
p {
  color: blue;
  font-size: 16px;
}
```

---

## Inline CSS

- **Definition**:
  Inline CSS ka use kisi specific HTML element ko style karne ke liye kiya jata hai. CSS ko directly HTML tag ke andar `style` attribute ke through likha jata hai.
- **Advantages**:
  - Simple aur quick styling for specific elements.
  - Specific styles define karne ke liye useful hai.
- **Disadvantages**:
  - Reusability nahi hoti.
  - Web page ka code cluttered aur less readable ho jata hai.
- **Example**:

```
<p style="color: red; font-size: 18px;">This is a paragraph with inline
CSS.</p>
```

---

## Internal CSS

- **Definition**:
  Internal CSS ek single HTML document ke liye styling provide karta hai. CSS code `<style>` tag ke andar, `<head>` section me likha jata hai.
- **Advantages**:

  - Ek hi document ke multiple elements ko style karna easy hai.
  - HTML aur CSS ek hi file me hoti hai, isliye separate file manage karne ki zarurat nahi.
- **Disadvantages**:
  - Ek file ke liye hi use hota hai, reusability nahi hoti.
  - Large projects me alag-alag HTML pages ke liye duplicate CSS likhni padti hai.
- **Example**:

```
<html>
<head>
  <style>
    body {
      background-color: lightblue;
    }
    h1 {
      color: green;
      font-family: Arial, sans-serif;
    }
  </style>
</head>
<body>
  <h1>This I    s a heading with internal CSS.</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

---

## External CSS

- **Definition**:
  External CSS ka use ek separate `.css` file me styling likhne ke liye kiya jata hai. HTML file me `<link>` tag ka use karke CSS file ko include kiya jata hai.
- **Advantages**:
  - Reusability: Ek hi CSS file multiple web pages me use ki ja sakti hai.
  - Maintainability: Styling aur HTML alag-alag hone se code clean aur manageable hota hai.
  - Better Performance: Browser CSS file ko cache kar sakta hai, jo loading time reduce karta hai.
- **Disadvantages**:
  - CSS aur HTML alag-alag hone ki wajah se debugging me time lagta hai.
  - Web page ka style external file load hone par hi apply hoga.
- **Example**:
  **External CSS File (`styles.css`)**:

```
body {
  background-color: lightgray;
}
h1 {
  color: darkblue;
  font-size: 24px;
}
```

**HTML File**:

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1>This is a heading with external CSS.</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

## Comparison Table of CSS

| Type | Location | Scope | Reusability | Ease of Maintenance |
|---|---|---|---|---|
| Inline CSS | Inside the HTML element | Specific to one element | No | Difficult |
| Internal CSS | Inside `<style>` in `<head>` | For one HTML document | No | Moderate |
| External CSS | In a separate `.css` file | Across multiple documents | Yes | Easy |

---

Chaliye **Bootstrap-CSS, Text, CSS Forms, aur CSS Components (Dropdown)** ko detail me samajhte hain hinglish me:

---

## Bootstrap-CSS

- **Definition**:
  Bootstrap ek popular **CSS framework** hai jo responsive aur visually appealing websites banane ke liye use hota hai.
  Yeh pre-designed CSS aur JavaScript components provide karta hai, jo web development ko fast aur efficient banata hai.
- **Key Features**:
  1. **Responsive Design**: Websites automatically har screen size (mobile, tablet, desktop) ke liye adjust ho jaati hain.
  2. **Predefined Classes**: CSS styling ke liye ready-to-use classes milti hain (e.g., `btn`, `container`, `row`, `col`).
  3. **Components**: Buttons, navbars, dropdowns, modals, and forms jaise reusable UI components provide karta hai.
  4. **Grid System**: 12-column layout system use karke layouts create karna easy hota hai.
- **How to Use Bootstrap**:
  Bootstrap ko include karne ke liye CDN ka use hota hai ya local files download karke project me add kiya ja sakta hai.
  **Example (CDN)**:

```
<head>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.mi
n.css">
</head>
```

---

## Text Styling in CSS and Bootstrap

- **CSS for Text Styling**:
  CSS ka use text ke appearance ko control karne ke liye hota hai (e.g., color, size, alignment).
  **Example**:

```
h1 {
  color: blue;
  text-align: center;
  font-size: 24px;
}
```

- **Bootstrap Text Utilities**:
  Bootstrap predefined text classes provide karta hai jo quick styling ke liye kaam aati hain.
  **Common Text Classes**:
  - `text-start`: Left align text.
  - `text-center`: Center align text.
  - `text-end`: Right align text.
  - `text-muted`: Light gray color text.
  - `fw-bold`: Bold text.
  - `fst-italic`: Italic text.

  **Example**:

```
<p class="text-center text-muted">This is centered and muted
text.</p>
```

---

## CSS Forms

- **Definition**:
  Forms ka use user input collect karne ke liye hota hai, aur CSS forms ko visually appealing banata hai.
- **CSS for Forms**:
  CSS ka use karke forms ke elements (input fields, buttons, etc.) ko style kiya jata hai.
  **Example**:

```
input[type="text"] {
  width: 100%;
  padding: 10px;
  margin: 5px 0;
  border: 1px solid #ccc;
  border-radius: 4px;
}
```

```
button {
  background-color: blue;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 4px;
}
```

- **Bootstrap Forms**:
  Bootstrap predefined classes provide karta hai jo forms ko quickly style karte hain.
  **Common Form Classes**:
  - `form-control`: Input fields ko style karne ke liye.
  - `form-group`: Form elements ko group karne ke liye.
  - `btn btn-primary`: Styled button banane ke liye.

  **Example**:

```
<form>
  <div class="mb-3">
    <label for="name" class="form-label">Name</label>
    <input type="text" class="form-control" id="name">
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

---

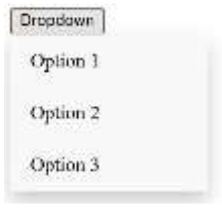
## CSS Components – Dropdown

- **Definition**:
  Dropdown ek UI component hai jo ek menu create karta hai jo tabhi visible hota hai jab user usko click ya hover kare.
  Dropdowns ka use navigation ya options ko compact form me show karne ke liye hota hai.
- **CSS for Dropdown**:
  CSS ka use karke basic dropdown menus banaye ja sakte hain.
  **Example**:

```
<style>
 .dropdown {
   position: relative;
   display: inline-block;
 }
 .dropdown-content {
   display: none;
   position: absolute;
   background-color: #f9f9f9;
   min-width: 160px;
   box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
   z-index: 1;
 }
 .dropdown:hover .dropdown-content {
   display: block;
```

```
  }
  .dropdown-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
  }
  .dropdown-content a:hover {
    background-color: #f1f1f1;
  }
</style>
<div class="dropdown">
  <button>Dropdown</button>
  <div class="dropdown-content">
    <a href="#">Option 1</a>
    <a href="#">Option 2</a>
    <a href="#">Option 3</a>
  </div>
</div>
```

**Output:**



- **Bootstrap Dropdowns**:
  Bootstrap dropdowns ko quickly banane ke liye predefined classes ka use hota hai.
  **Classes for Dropdown**:
  - `dropdown`: Dropdown container.
  - `dropdown-toggle`: Button ya link ke liye jo dropdown open kare.
  - `dropdown-menu`: Dropdown ke options.
  - `dropdown-item`: Har ek item ke liye.

  **Example**:

```
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button" data-bs-
toggle="dropdown">
    Dropdown
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Option 1</a></li>
    <li><a class="dropdown-item" href="#">Option 2</a></li>
    <li><a class="dropdown-item" href="#">Option 3</a></li>
  </ul>
</div>
```

## Introduction of CSS3

CSS3 (Cascading Style Sheets, Level 3) CSS ka latest version hai jo websites aur web applications ka **look and feel** design karne ke liye use hota hai. Isme naye features aur properties add kiye gaye hain jo web designing ko advanced aur dynamic banate hain.

**Key Features of CSS3:**

1. **Advanced Styling**:
   - New properties for text effects, box shadows, and gradients.
   - Rounded corners using `border-radius`.
2. **Animation and Transition**:
   - **Animation**: Elements ko move ya animate karne ke liye.

```
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
div {
  animation-name: example;
  animation-duration: 4s;
}
```

   - **Transition**: Smooth changes between states.

```
div {
  transition: background-color 2s;
}
div:hover {
  background-color: blue;
}
```

3. **Media Queries**:
   - Responsive design ke liye different screen sizes ko target karna.

```
@media (max-width: 600px) {
          body {
            background-color: lightblue;
          }
        }
```

4. **New Selectors**:
   - Select elements with more precision. Example:
     - `nth-child()`
     - `last-of-type`
5. **Flexbox and Grid**:
   - Layout banane ke liye powerful tools.
   - Flexbox for row/column alignment.

```
display: flex;
justify-content: center;
```

```
align-items: center;
```

6. **Web Fonts**:
   - Online fonts (e.g., Google Fonts) ko import kar ke use karna.

**Advantages of CSS3:**

- Lightweight and fast loading.
- Improved user experience with animations and transitions.
- Makes websites responsive and adaptive for all devices.

---

## Bootstrap

Bootstrap ek **CSS Framework** hai jo **HTML, CSS, aur JavaScript** ke ready-made components ka set provide karta hai. Iska use karke aap fast aur visually appealing websites bana sakte hain.

**Introduction:**

- Developed by **Twitter** engineers in 2011.
- Latest version: Bootstrap 5.
- Focus: **Responsive Design**, **Mobile-First Approach**, and **Predefined Styles**.

**Key Features of Bootstrap:**

1. **Grid System**:
   - 12-column responsive grid layout.

```
<div class="container">
  <div class="row">
    <div class="col-6">Column 1</div>
    <div class="col-6">Column 2</div>
  </div>
</div>
```

2. **Predefined Components**:
   - **Navbar**: Navigation menu.
   - **Cards**: Display content like articles, images.
   - **Buttons**: Styled buttons with predefined classes (`btn`, `btn-primary`).
3. **Responsive Design**:
   - Websites automatically adjust for devices like mobiles, tablets, and desktops.
4. **Utility Classes**:
   - Easy customization using pre-built classes.
   - Example: `text-center`, `bg-light`, `p-3`.
5. **JavaScript Components**:
   - Dropdowns, modals, tooltips, and carousels.

**Advantages of Bootstrap:**

- Saves time with prebuilt components.
- Highly customizable using custom CSS.
- Compatible with all modern browsers.

**Example of Bootstrap Usage:**

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
  <div class="container text-center">
    <h1 class="text-primary">Welcome to Bootstrap!</h1>
    <button class="btn btn-success">Click Me</button>
  </div>
</body>
</html>
```

## Client-Side Programming: JavaScript Language

JavaScript ek **client-side programming language** hai jo websites aur web applications ko dynamic aur interactive banata hai. Client-side ka matlab hai ki code directly browser (user ke computer) par run hota hai, na ki server par.

### JavaScript Kya Hai?

- Ek lightweight, interpreted scripting language hai.
- Web pages mein logic add karne ke liye use hoti hai.
- **HTML** aur **CSS** ke sath integrate karke interactive features create karte hain, jaise sliders, forms validation, pop-ups, etc.

---

### JavaScript Ke Features:

1. **Lightweight**:
    - Browser me directly run hoti hai, kisi server ka load nahi hota.
2. **Event-Driven Programming**:
    - User ke actions ke response me kama kar sakta hai (e.g., button click, hover).
3. **Dynamic Content**:
    - Web page ke content ko bina reload kare update kar sakta hai.
4. **Cross-Browser Support**:

  - Har modern browser (Chrome, Firefox, Safari) ke sath compatible hai.
5. **Built-In Functions**:
  - Date, time, math calculations, etc. ke liye predefined functions available hain.

---

## Use of Javascript

JavaScript ko website ke **behavior** aur **functionality** ke liye use kiya jata hai.

**Example:**

1. **Form Validation**:
  - User form submit karte samay, data sahi format me hai    an ahi ye check karta hai.
2. **Interactive UI**:
  - Button clicks, image sliders, modals, etc. create karta hai.
3. **Asynchronous Programming**:
  - APIs ke sath data exchange kar ke page ko reload kiye bina update karta hai.

---

## JavaScript Syntax

**Hello World Example:**

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Example</title>
</head>
<body>
  <h1 id="demo">Hello!</h1>
  <button onclick="changeText()">Click Me</button>

  <script>
    function changeText() {
      document.getElementById("demo").innerHTML = "Hello, JavaScript!";
    }
  </script>
</body>
</html>
```

**Explanation**:

- Ek button par click karte hi, `<h1>` element ka text dynamically update ho jata hai.

---

## JavaScript Ke Core Concepts:

1. **Variables**: Data ko store karne ke liye.

```
let name = "Kundan";
console.log(name);
```

**Output:** Kundan

2. **Functions**: Reusable code blocks.

```
function greet() {
  console.log("Welcome to JavaScript!");
}
greet();
```

**Output:** Welcome to JavaScript!

3. **Events**: User actions handle karte hain.

```
document.querySelector("button").addEventListener("click", function() {
  alert("Button clicked!");
});
```

4. **DOM Manipulation**:
   - HTML elements ko JavaScript se control karna.

```
document.getElementById("demo").style.color = "blue";
```

---

## Advantages of JavaScript in Client-Side Programming:

1. **Fast Execution**:
   - Browser par directly run hoti hai, isliye fast hoti hai.
2. **Reduces Server Load**:
   - Server-side tasks jaise form validation ya simple calculations ko browser me handle karta hai.
3. **Rich User Experience**:
   - Interactive aur dynamic UI elements provide karta hai.
4. **Versatile**:
   - Modern frameworks jaise **React**, **Vue**, aur **Angular** ke sath use karke advanced web apps banayi ja sakti hain.

### JavaScript Objects

JavaScript objects ek tarike ka **data structure** hain jo key-value pairs ka collection hote hain. Iska use data ko logically group karne aur methods ko store karne ke liye hota hai.

**Example**

```
let student = { name: "Kundan", age: 21 };
console.log(student.name);
```

**Output:** Kundan

**{Kundan Bharti}**

## JavaScript Alert

`alert()` ek simple popup box hai jo message dikhata hai. Ye mostly warnings ya information dene ke liye use hota hai.

**Example**

```
alert("This is an alert!");
```

---

## JavaScript Button

Button par user ke action (click) ko handle karne ke liye JavaScript ka use hota hai. Ye kisi specific task ko trigger kar sakta hai.

**Example (Short Code):**

```
<button onclick="alert('Button clicked!')">Click Me</button>
```

---

## JavaScript Popover

Popover ek chhota popup box hai jo extra information show karta hai. Ye mostly **Bootstrap framework** ke sath use hota hai.

**Example**

```
<button type="button" data-bs-toggle="popover" data-bs-content="Popover content">
  Hover Me
</button>
```

---

## Host Objects

Host objects wo objects hote hain jo browser ya JavaScript runtime environment (e.g., Node.js) provide karta hai. Ye JavaScript language ke core part nahi hote, lekin functionality add karte hain.

**Examples:**

- **Window Object**: Represent karta hai browser window.
- **Document Object**: HTML document ko represent karta hai.
- **Navigator Object**: Browser ke details provide karta hai (e.g., browser name, version).
- **Console Object**: Debugging ke liye logs dikhata hai.

**Key Point:**

Host objects browser-specific hote hain, aur unka behavior har browser me thoda alag ho sakta hai.

**Browsers**

Browsers JavaScript ko execute karte hain aur HTML, CSS ke sath integrate karte hain. Browser ke main components:

1. **Rendering Engine**:
   - HTML aur CSS ko parse karke web page render karta hai.
   - Example: Blink (Chrome, Edge), WebKit (Safari).
2. **JavaScript Engine**:
   - JavaScript code ko execute karta hai.
   - Example: V8 (Chrome), SpiderMonkey (Firefox).
3. **APIs**:
   - Host objects ko access karne ke liye browsers APIs provide karte hain.
   - Example: fetch(), localStorage.

---

**DOM (Document Object Model)**

DOM ek programming interface hai jo HTML aur XML documents ko represent karta hai. DOM document ko ek **tree structure** ke form me dikhata hai jaha har element ek node hota hai.

**Key Features of DOM:**

1. **Dynamic Content Manipulation**:
   - DOM ki help se web page ke elements ko dynamically update karte hain.
2. **Tree Structure**:
   - Root node: `<html>`.
   - Child nodes: `<head>`, `<body>`, etc.
3. **Events**:
   - DOM events jaise `click`, `hover` ko handle karke interactive behavior add kiya jata hai.

---

## Example: DOM Manipulation (Short Code)

```
<!DOCTYPE html>
<html>
<body>
  <h1 id="title">Old Title</h1>
  <button onclick="changeTitle()">Change Title</button>
  <script>
    function changeTitle() {
      document.getElementById("title").innerHTML = "New Title";
    }
  </script>
</body>
</html>
```

**{Kundan Bharti}**

- **Explanation**: Button click karne par `<h1>` ka text dynamically change ho jata hai.

---

## Server-Side Programming

### 1. Java Servlets: Introduction

Java Servlets ek **server-side technology** hai jo dynamic web content generate karne ke liye use hoti hai. Servlet ek Java class hai jo web server ke request aur response ko handle karta hai.

**Features of Servlets:**

- **Platform Independent**: Java-based, toh alag-alag servers par kaam karte hain.
- **Efficient**: Lightweight aur multithreaded hote hain.
- **Secure**: Java ke built-in security features ka fayda lete hain.
- **Dynamic Content**: User input ke basis par responses generate karte hain.

---

### 2. Basics of Java Servlets

1. **How Servlets Work**:
   - User browser se request bhejta hai (HTTP Request).
   - Web server request ko Servlet ke pass forward karta hai.
   - Servlet process karta hai aur response web server ko bhejta hai.
   - Web server response browser ko send karta hai.
2. **Servlet Lifecycle**:
   - **`init()`**: Servlet initialize hota hai.
   - **`service()`**: Request process karta hai aur response generate karta hai.
   - **`destroy()`**: Servlet ko memory se remove kar diya jata hai.

---

## Simple Program

**Example: Basic Servlet Code**

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>Welcome to Java Servlets</h1>");
    }
}
```

**Explanation:**

- `doGet`: HTTP GET request ko handle karta hai.
- `response.getWriter()`: Response ke liye output stream provide karta hai.
- **Output**: Browser me "Welcome to Java Servlets" dikhega.

---

## Separating Programming and Presentation

Java Servlets me programming logic aur presentation (HTML) ko alag karne ka best practice hai. Isse code zyada organized aur maintainable hota hai.

**Approach:**

1. **Servlets for Logic**: Data ko process karte hain aur results generate karte hain.
2. **JSP/HTML for Presentation**: HTML format me result display karte hain.

### Example:

- **Servlet Code**:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class DataServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        request.setAttribute("message", "Hello from Servlet!");
        RequestDispatcher rd = request.getRequestDispatcher("display.jsp");
        rd.forward(request, response);
    }
}
```

- **JSP Code (`display.jsp`)**:

```
<html>
<body>
  <h1>${message}</h1>
</body>
</html>
```

**Explanation:**

- Servlet logic ko handle karta hai aur JSP ke pass result forward karta hai.
- JSP sirf data ko present karta hai.

---

# Unit-III ASP/JSP

**ASP/JSP (Active Server Pages / JavaServer Pages)**

**ASP (Active Server Pages)** aur **JSP (JavaServer Pages)** dono server-side scripting languages hain, lekin dono alag technologies hain. In dono ka use dynamic web pages banane ke liye hota hai, jisme server ko request bhejne par HTML content dynamically generate hota hai.

- **ASP**: Yeh Microsoft ka ek technology hai. ASP server-side scripting language hai, jisme aap server par logic likhte ho aur client ko dynamic HTML generate karke bhejte ho. ASP ka use mainly **VBScript** ya **JScript** ke saath hota hai.
- **JSP**: Yeh Java-based technology hai, jo HTML pages ko Java code ke saath dynamically generate karta hai. JSP ka use server par Java code likhne ke liye hota hai, aur yeh client ko HTML ke form mein output bhejta hai. JSP mein Java language ka use hota hai, jo scalable aur robust hoti hai.

**ASP** ka use mainly **Windows-based servers** pe hota hai, jabki **JSP** ka use **Java-based servers** pe hota hai, jaise **Apache Tomcat**.

---

## JSP Basics

**JSP** (JavaServer Pages) ek technology hai jiska use dynamic web pages banane ke liye hota hai. JSP ka basic idea hai ki Java code ko **HTML** ke andar embed kiya jaye, taki server dynamic content generate kar sake.

**JSP Page ka Structure:**

1. **Directives**: Yeh page ke configuration settings define karte hain, jaise page encoding ya language. Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" %>
```

2. **Declarations**: Yeh Java variables aur methods ko define karte hain jo page me use honge. Example:

```
<%! int counter = 0; %>
```

3. **Scriptlets**: Yeh Java code likhne ke liye use hota hai jo HTML ke andar execute hota hai. Yeh `<% %>` ke andar likha jata hai. Example:

```
<%
  counter++;
  out.println("Counter: " + counter);
%>
```

4. **Expressions**: Yeh Java code ko HTML output ke roop me display karta hai. Example:

```
<%= "Hello, World!" %>
```

5. **Comments**: JSP me comments ko `<!-- -->` ya `<%-- --%>` ke beech likha jata hai.

**JSP Ka Working:**

1. Client se request aati hai JSP page ke liye.
2. Server JSP page ko compile karta hai Java servlet me.
3. Java servlet ko execute karke dynamic content generate kiya jata hai.
4. Server client ko HTML response bhejta hai.

---

## ASP/JSP Objects

ASP aur JSP dono mein kuch predefined objects hote hain jo aapko dynamic content generate karte waqt use karne padte hain. In objects ko server-side scripting me access kiya jata hai.

**ASP Objects:**

1. **Request Object**:
   - Yeh object client se server tak aane wali request ki information store karta hai.
   - Aap is object ka use form data ya URL parameters ko fetch karne ke liye karte ho.
   - **Example:**

   ```
   <%
     Dim userName
     userName = Request.QueryString("username")
     Response.Write("Hello, " & userName)
   %>
   ```

   - **Request.Form** aur **Request.QueryString** ka use karke data retrieve kiya jata hai.
2. **Response Object**:
   - Yeh object server se client tak response bhejne ke liye use hota hai.
   - Aap is object ka use HTML content ko generate karne ke liye karte ho.
   - **Example:**

   ```
   <%
     Response.Write("Welcome to ASP!")
   %>
   ```

   - **Response.Redirect** ka use kisi dusre page pe redirect karne ke liye hota hai.
3. **Session Object**:
   - Yeh object ek user session ko track karta hai. Iska use user-specific information store karne ke liye hota hai, jaise login information.
   - **Example:**

   ```
   <%
     Session("username") = "Kundan"
     Response.Write(Session("username"))
   %>
   ```

4. **Application Object**:
   - Yeh object global level par data ko store karta hai, jo application ke lifetime tak available rehta hai.
   - **Example:**

```
<%
  Application("appName") = "ASP Web App"
  Response.Write(Application("appName"))
%>
```

**JSP Objects:**

1. **Request Object**:
   - Yeh object HTTP request ko represent karta hai, jisme client ke dwara bheje gaye data ko store kiya jata hai.
   - **Example:**

```
<%
  String username = request.getParameter("username");
  out.println("Welcome, " + username);
%>
```

2. **Response Object**:
   - Yeh object HTTP response ko represent karta hai. Iska use client ko HTML response bhejne ke liye hota hai.
   - **Example:**

```
<%
  response.setContentType("text/html");
  out.println("Welcome to JSP!");
%>
```

3. **Session Object**:
   - Yeh object session ke data ko store karta hai. Jaise agar user login hota hai, toh session me user ka data store kar sakte hain.
   - **Example:**

```
<%
  session.setAttribute("username", "Kundan");
  out.println("User: " + session.getAttribute("username"));
%>
```

4. **Application Object**:
   - Yeh object application-wide data ko store karta hai. Yeh object application ke puri lifecycle tak data ko hold karta hai.
   - **Example:**

```
<%
  getServletContext().setAttribute("appName", "JSP Web App");
  out.println("App Name: " + getServletContext().getAttribute("appName"));
%>
```

## Simple ASP/JSP Pages

**ASP Page** aur **JSP Page** dono ka use dynamic content generate karne ke liye hota hai. Dono me JavaScript ya server-side code ko HTML ke andar embed karke client ke browser ko dynamic content bheja jata hai.

**ASP Page Example:**

ASP page me server-side scripting ka use hota hai jisme **VBScript** ya **JScript** likhte hain. Server ko request bhejne par yeh dynamic content generate karta hai.

**ASP Page ka Structure**:

```
<html>
  <body>
    <%
      ' Server-side code likha gaya hai
      Dim username
      username = "Kundan"
      Response.Write("Hello, " & username)
    %>
  </body>
</html>
```

- **Explanation**: Is example me, `Response.Write` se server "Hello, Kundan" print karega jab page request kiya jayega.

**JSP Page Example:**

JSP page me Java code ko HTML ke andar likha jata hai. JSP me **Java language** ka use hota hai.

**JSP Page ka Structure**:

```
<html>
  <body>
    <%
      // Server-side Java code
      String username = "Kundan";
      out.println("Hello, " + username);
    %>
  </body>
</html>
```

- **Explanation**: Yeh page request hone par `out.println` ke through server "Hello, Kundan" print karega.

**Difference**: ASP me VBScript/JScript hota hai, aur JSP me Java language ka use hota hai.

## Representing Web Data

**Web Data** ko represent karne ka matlab hai ki client ko server se data dynamically bhejna. Yeh data **databases** se ya kisi external source se aa sakta hai, aur server us data ko **HTML** ya **XML** format me client ko bhejta hai.

**ASP Example (Representing Data):**

Agar server ko kisi database se data fetch karna ho, toh us data ko HTML ke andar dikhaya jaata hai.

**ASP Page Example**:

```
<%
  Dim conn, rs, query
  Set conn = Server.CreateObject("ADODB.Connection")
  conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\path\to\your\database.mdb"

  query = "SELECT username FROM users"
  Set rs = conn.Execute(query)

  Do While Not rs.EOF
    Response.Write("<p>" & rs("username") & "</p>")
    rs.MoveNext
  Loop

  rs.Close
  conn.Close
%>
```

- **Explanation**: Is example me, database se `username` fetch kiya gaya hai aur har `username` ko HTML page me dynamically dikhaya gaya hai.

**JSP Example (Representing Data):**

JSP me bhi same process hota hai, bas aapko **JDBC** ka use karke database se data fetch karna padta hai.

**JSP Page Example**:

```
<%@ page import="java.sql.*" %>
<%
  Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root",
"password");
  Statement stmt = conn.createStatement();
  ResultSet rs = stmt.executeQuery("SELECT username FROM users");

  while (rs.next()) {
    out.println("<p>" + rs.getString("username") + "</p>");
  }
  rs.close();
  stmt.close();
  conn.close();
%>
```

- **Explanation**: Is example me, `username` ko JDBC ka use karke database se fetch kiya gaya hai aur HTML page me display kiya gaya hai.

---

## Database Connectivity

**Database Connectivity** ka matlab hai ki aapke ASP ya JSP page ko database se connect karna taaki aap data fetch kar sakein, update kar sakein ya insert kar sakein.

**ASP Database Connectivity (Using ADO):**

ASP me **ADO (ActiveX Data Objects)** ka use hota hai data ko retrieve karne ke liye.

1. **Connection**: Aapko database se connect karne ke liye `Connection` object ka use karna padta hai.
2. **Recordset**: Aapko data ko fetch karne ke liye `Recordset` object ka use hota hai.

**ASP Example (Database Connectivity)**:

```
<%
  Dim conn, rs
  Set conn = Server.CreateObject("ADODB.Connection")
  conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\path\to\your\database.mdb"

  Set rs = conn.Execute("SELECT * FROM users")
  While Not rs.EOF
    Response.Write(rs("username") & "<br>")
    rs.MoveNext
  Wend

  rs.Close
  conn.Close
%>
```

- **Explanation**: Is code me, `ADODB.Connection` object ka use karke ASP page database se data fetch karta hai aur `Response.Write` se browser me display karta hai.

**JSP Database Connectivity (Using JDBC):**

JSP me **JDBC** ka use hota hai data ko database se connect karne aur retrieve karne ke liye.

1. **Connection**: JDBC driver se database connection establish hota hai.
2. **Statement**: SQL queries ko execute karne ke liye `Statement` object ka use hota hai.
3. **ResultSet**: ResultSet me database ka data store hota hai.

**JSP Example (Database Connectivity)**:

```
<%@ page import="java.sql.*" %>
<%
  Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root",
"password");
  Statement stmt = conn.createStatement();
  ResultSet rs = stmt.executeQuery("SELECT * FROM users");

  while (rs.next()) {
    out.println(rs.getString("username") + "<br>");
  }

  rs.close();
  stmt.close();
  conn.close();
%>
```

- **Explanation**: JSP me JDBC ka use karte hue database connection establish hota hai aur data ko **ResultSet** se fetch karke HTML page me display kiya jata hai.

---

## 1. JDBC (Java Database Connectivity)

**JDBC** (Java Database Connectivity) ek API hai jo Java applications ko databases ke saath connect karne ke liye use hoti hai. JDBC aapko SQL queries ko execute karne aur database se data fetch ya manipulate karne ki suvidha deta hai. JDBC ko use karne ke liye, aapko pehle apne database ke liye driver configure karna padta hai, phir **Connection** object banake **Statement** ka use karke SQL queries execute karte hain, aur results ko **ResultSet** mein store karte hain.

**Steps**:

1. **Establish Connection**: `Connection` object ka use karke database se connection establish kiya jata hai.
2. **Execute SQL Queries**: `Statement` object ka use karke SQL queries execute ki jaati hain.
3. **Process the Results**: **ResultSet** object mein query ka result store hota hai.
4. **Close the Connection**: Resources ko release karne ke liye connection ko close kiya jata hai.

**Short Code Example (JDBC)**:

```
import java.sql.*;
public class JDBCExample {
  public static void main(String[] args) {
    try {
      Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root",
"password");
      Statement stmt = conn.createStatement();
      ResultSet rs = stmt.executeQuery("SELECT * FROM users");
      while (rs.next()) System.out.println(rs.getString("username"));
      rs.close(); stmt.close(); conn.close();
    } catch (Exception e) { e.printStackTrace(); }
  }
}
```

---

## 2. Dynamic Web Pages

**Dynamic Web Pages** wo pages hote hain jo user ki request ke hisaab se content ko generate karte hain. Yani, har baar page reload hone par page ka content change hota hai. Dynamic pages server-side scripting languages jaise **ASP**, **JSP**, **PHP**, ya **Node.js** se generate kiye jaate hain. Yeh pages real-time data ko show karte hain, jaise ki database se information retrieve karna ya user input ko process karna.

**Example**: JSP page ya ASP page me, hum user ka input ya database data ko dynamically display karte hain.

**Short Code Example (JSP)**:

```
<%@ page import="java.sql.*" %>
<%
  Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/mydb", "root",
"password");
  Statement stmt = conn.createStatement();
  ResultSet rs = stmt.executeQuery("SELECT * FROM users");
  while (rs.next()) {
    out.println("<p>" + rs.getString("username") + "</p>");
  }
  rs.close(); stmt.close(); conn.close();
%>
```

---

## 3. XML (eXtensible Markup Language)

**XML** ek markup language hai jo data ko structured format mein store karti hai. Iska use mostly data interchange ke liye hota hai, jahan par data ko ek standardized format mein send aur receive kiya jata hai. XML mein data ko **tags** ke andar define kiya jata hai, aur har tag ke andar ek specific value hoti hai.

**Example**: Aap kisi person ka naam, age, aur city ko XML format me define kar sakte hain.

**Short Code Example (XML)**:

```
<person>
  <name>Kundan</name>
  <age>22</age>
  <city>Bhopal</city>
</person>
```

## 4. DTD (Document Type Definition)

**DTD** XML document ke structure ko define karta hai. Iska kaam hota hai XML document ko validate karna. Aap DTD ke through specify kar sakte hain ki kis type ka data kis tag ke andar hona chahiye. Yeh **internal** ya **external** ho sakta hai.

**Example**: DTD define karta hai ki **person** element mein **name**, **age**, aur **city** hone chahiye aur unke andar text (PCDATA) hoga.

**Short Code Example (DTD)**:

```
<!DOCTYPE person [
  <!ELEMENT person (name, age, city)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT age (#PCDATA)>
  <!ELEMENT city (#PCDATA)>
]>
<person>
  <name>Kundan</name>
  <age>22</age>
  <city>Bhopal</city>
</person>
```

---

## 5. XML Schema

**XML Schema** ek powerful tool hai jo XML document ke structure ko define karta hai. Yeh DTD se zyada detailed hota hai aur aapko type checking aur data validation ki zyada flexibility deta hai. XML Schema ka use karke hum data types define kar sakte hain, jaise **string**, **integer**, **date**, etc.

**Example**: XML Schema ko use karke hum XML document ke structure ko validate kar sakte hain.

**Short Code Example (XML Schema)**:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="person">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="age" type="xs:int"/>
        <xs:element name="city" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

---

## 6. DOM (Document Object Model)

**DOM** (Document Object Model) ek programming interface hai jo HTML aur XML documents ko access aur modify karne ke liye use hota hai. DOM ko use karke hum document ko ek tree-like structure mein represent karte hain, jisme har element ek node hota hai. Isse hum programmatically HTML ya XML document ke elements ko manipulate kar sakte hain.

**Example**: Aap DOM ka use karke XML document ke kisi bhi element ko read ya modify kar sakte hain.

**Short Code Example (DOM)**:

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
public class DOMExample {
  public static void main(String[] args) {
    try {
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder builder = factory.newDocumentBuilder();
      Document doc = builder.parse("person.xml");
      NodeList list = doc.getElementsByTagName("name");
      System.out.println(list.item(0).getTextContent());
    } catch (Exception e) { e.printStackTrace(); }
  }
}
```

---

## 1. Building Web Applications

**Web applications** wo applications hote hain jo **web browser** ke through accessible hote hain. Inme client-side (frontend) aur server-side (backend) hota hai. Web applications ka main purpose user ko interact karne ki suvidha dena hota hai, jaise online shopping, social media platforms, etc. **Frontend** mein HTML, CSS, aur JavaScript ka use hota hai, jabki **backend** mein server-side technologies jaise PHP, JSP, ASP, Node.js, etc., use hoti hain.

- **Frontend** (Client-Side): Frontend wo part hota hai jo user directly interact karta hai. Yeh page ka design, layout, aur elements ko manage karta hai. Isme HTML (structure), CSS (styling), aur JavaScript (interactivity) ka use hota hai.
- **Backend** (Server-Side): Backend wo part hai jo server pe execute hota hai. Yeh application ka logic, database operations, aur server-side processing handle karta hai. Backend ke liye PHP, Java, Python, Node.js, etc., use kiye jaate hain.
- **Database**: Web applications ko data store karne ke liye databases ki zarurat hoti hai. **SQL** ya **NoSQL** databases (jaise MySQL, MongoDB) ko backend ke sath integrate kiya jata hai.

- **Architecture**: Web applications usually **client-server architecture** follow karti hain, jisme client request bhejta hai aur server response provide karta hai.

---

## 2. Cookies

**Cookies** small pieces of data hote hain jo web browser me store kiye jaate hain aur jab bhi user server se request bhejta hai, to yeh data server ko bheja jaata hai. Cookies ka use mainly **session management**, **user preferences**, aur **tracking** ke liye hota hai.

- **Purpose**: Cookies user-specific hoti hain aur web application ke behavior ko personalize karti hain. Agar user kisi site pe login hota hai, toh cookie uska login state store kar sakti hai, taaki wo har page par dobara login na kare.
- **Working**: Jab user browser me koi website visit karta hai, to web server ek cookie generate karta hai aur usse user ke browser me send karta hai. Browser wo cookie apne local storage me store karta hai aur har request ke sath server ko bhejta hai.
- **Types of Cookies**:
    1. **Session Cookies**: Yeh cookies temporary hoti hain aur browser band hone par delete ho jaati hain.
    2. **Persistent Cookies**: Yeh cookies ek specific time period ke liye store hoti hain aur user ke next visit ke liye bhi valid rehti hain.
- **Security Considerations**: Cookies ko secure rakhne ke liye **secure flag** aur **HTTP-only flag** set kiya jata hai. Yeh prevent karta hai cookies ko unauthorized access se.

---

## 3. Sessions

**Sessions** ek mechanism hai jo user ke data ko track karne ke liye use kiya jata hai jab wo multiple pages visit karta hai. Sessions server-side pe data store karte hain, jo ek specific user ke liye hota hai.

- **Purpose**: Jab user web application ko visit karta hai, to server ek unique **session ID** generate karta hai, jo subsequent requests ke sath bheja jata hai. Yeh session ID user ki activities ko track karta hai, jaise login status, user preferences, etc.
- **How Sessions Work**: Jab user ek page visit karta hai, to server ek session ID generate karta hai aur browser ko send karta hai. Jab bhi user koi aur page visit karta hai, to session ID server ke paas bheja jata hai, aur server user ka data retrieve karta hai.
- **Session Storage**: Session data server-side pe store hota hai aur client-side pe **cookies** ke through track kiya jata hai. Session end hone par data destroy ho jata hai.

- **Use Case**: Login pages mein sessions ka use hota hai, jahan server user ka login status track karta hai. Agar user login kar chuka hai, to session ID ke zariye uska session validate hota hai aur wo application ke har page tak access kar sakta hai.

---

**Difference in Cookies and Sessions**

| Feature | Cookies | Sessions |
|---|---|---|
| **Storage Location** | Client-side (browser mein store hoti hain) | Server-side (server par store hoti hain) |
| **Data Storage Capacity** | Limited (around 4KB per cookie) | Larger capacity (session data server pe store hoti hai) |
| **Lifetime** | Can be persistent (expiry date set) or temporary (session cookies) | Temporary (session data expires when browser or session ends) |
| **Security** | Less secure, as they are stored on the client-side | More secure, as data is stored on the server-side |
| **Access** | Data can be accessed by both client and server | Data is only accessed by the server |
| **Performance Impact** | Minimal impact on performance | Can increase server load if there are many concurrent sessions |
| **Use Case** | User preferences, login states, tracking information | User authentication, tracking user state across multiple pages |
| **Expiry** | Can be set (persistent cookies) or last until browser is closed (session cookies) | Expires when the session ends (browser closes or session timeout) |
| **Common Usage** | Remembering login, preferences, shopping cart information | User authentication, cart tracking, login session management |

## 4. Open Source Environment

**Open Source Environment** ka matlab hai wo software jo publicly available hota hai aur jiska source code freely accessible hota hai. Open-source software ko log modify kar sakte hain, distribute kar sakte hain, aur apne needs ke according use kar sakte hain. Open source community ke log milke software ko improve karte hain, bugs fix karte hain, aur naye features add karte hain.

- **Characteristics of Open Source**:
  1. **Free to Use**: Open-source software ko aap bina kisi license fee ke use kar sakte hain.
  2. **Accessible Code**: Iska source code publicly available hota hai. Aap chahe to is code ko modify bhi kar sakte hain.

3. **Community-driven**: Open-source projects usually **community-driven** hote hain, jahan developers worldwide contribute karte hain.

- **Benefits of Open Source**:
  1. **Customization**: Aap apne requirements ke hisaab se open-source software ko customize kar sakte hain.
  2. **Security**: Open-source software ko open community review karti hai, isliye bugs aur vulnerabilities jaldi identify ho jaati hain.
  3. **Cost-effective**: Open-source software free hota hai, jo development aur usage ki cost ko kam karta hai.
- **Popular Open Source Software**:
  1. **Linux**: Ek open-source operating system hai jo widely use hota hai.
  2. **Apache HTTP Server**: Open-source web server software hai.
  3. **MySQL**: Ek open-source relational database management system (RDBMS) hai.
  4. **WordPress**: Open-source content management system (CMS) hai jo blogs aur websites create karne ke liye use hota hai.
- **Open Source Licensing**: Open source software ko use karne ke liye kuch specific licenses hote hain, jaise **GPL**, **MIT License**, etc., jo specify karte hain ki aap software kaise use, modify, aur distribute kar sakte hain.

# Unit-IV PHP And Mysql

## 1. Basic PHP Programs

**PHP (Hypertext Preprocessor)** ek server-side scripting language hai jo mainly dynamic web pages banane ke liye use hoti hai. PHP ka use aap websites par user input ko process karne, data store karne (jaise databases), aur interactive web applications banane ke liye karte ho. PHP code ko server pe execute kiya jata hai aur browser par output display hota hai.

**Basic PHP Syntax**: PHP code ko `<?php` aur `?>` tags ke andar likhte hain.

**Example:**

```
<?php
  echo "Hello, World!";
?>
```

**Explanation**:

- `<?php` se PHP code start hota hai.
- `echo` command se output print hota hai.
- `?>` se PHP code close hota hai.

**Example**:
Ek basic PHP program jo user ka naam display karta hai:

```
<?php
  $name = "Kundan";
  echo "Hello, " . $name . "!";
?>
```

**Explanation**:

- `$name` ek variable hai jisme "Kundan" store hai.
- `echo` se message display hota hai.

---

## 2. Form & PHP

**Forms** web pages par wo input fields hote hain jahan se user apni information submit karta hai. **PHP** ko form ke data ko process karne ke liye use kiya jata hai. Jab user form ko submit karta hai, to us data ko PHP script process karti hai, jisme data ko **GET** ya **POST** method se access kiya jata hai.

**Form Elements**:

- **Input**: Text fields, password fields, checkboxes, etc.

- **Submit**: Form ko submit karne ke liye button.
- **Select**: Drop-down list.
- **TextArea**: Multi-line text box.

**Form ka basic structure**:

```
<form action="process.php" method="post">
  Name: <input type="text" name="username"><br>
  Email: <input type="email" name="email"><br>
  <input type="submit" value="Submit">
</form>
```

**Explanation**:

- `action="process.php"` ka matlab hai jab form submit hoga, to data `process.php` page ko bheja jayega.
- `method="post"` ka matlab hai form data ko server ke through POST method se bheja jayega.

---

## 3. Creating Form Controls

**Form Controls** wo elements hote hain jo user input lene ke liye use hote hain. Form controls mein **input fields**, **radio buttons**, **checkboxes**, **select boxes**, aur **submit buttons** aate hain. PHP ke through hum form se values fetch karke unhe process kar sakte hain.

**Form Controls Examples**:

1. **Text Input**:

```
<input type="text" name="username" placeholder="Enter your name">
```

2. **Password Input**:

```
<input type="password" name="password" placeholder="Enter your password">
```

3. **Radio Buttons** (Single choice selection):

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

4. **Checkboxes** (Multiple choice selection):

```
<input type="checkbox" name="interests[]" value="music"> Music
<input type="checkbox" name="interests[]" value="sports"> Sports
```

5. **Dropdown (Select Box)**:

```
<select name="country">
  <option value="india">India</option>
  <option value="us">United States</option>
</select>
```

6. **Submit Button**:

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

**Explanation**:

- Har form control ka apna specific purpose hota hai, jaise text input se text enter karna, radio buttons se ek option select karna, checkboxes se multiple options select karna, etc.

---

## 4. Using Values Returned from Forms Using PHP

Jab form submit hota hai, to form ke data ko **PHP** ke through process kiya jata hai. Form data ko access karne ke liye PHP mein `$_GET` aur `$_POST` superglobals ka use hota hai. **`$_POST`** method data ko securely bhejta hai aur **`$_GET`** data ko URL ke through send karta hai.

**Using `$_POST` (for secure data handling)**:

```
<?php
  if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $email = $_POST['email'];
    echo "Welcome, " . $username . "! Your email is " . $email;
  }
?>
```

**Explanation**:

- `$_POST['username']` aur `$_POST['email']` ke through form data ko fetch kiya jaata hai.
- `$_SERVER["REQUEST_METHOD"]` se check kiya jata hai ki form POST method se submit hua hai ya nahi.

**Using `$_GET`**:

```
<?php
  if ($_SERVER["REQUEST_METHOD"] == "GET") {
    $username = $_GET['username'];
    $email = $_GET['email'];
    echo "Welcome, " . $username . "! Your email is " . $email;
  }?>
```

**Explanation**:

- **$_GET** method data ko URL ke part ke through send karta hai. Is method mein data visible hota hai URL ke andar, isliye sensitive information ke liye **POST** method better hota hai.

---

## PHP Database Connectivity

**PHP** ko **MySQL database** ke saath connect karne ke liye, hum **mysqli** (MySQL Improved) extension ka use karte hain. Iske through hum PHP se database ke saath interaction karte hain—jaise data insert karna, fetch karna, update karna, aur delete karna. **PDO (PHP Data Objects)** bhi ek alternative method hai, jo zyada flexible aur secure hota hai, lekin yahan hum **mysqli** par focus karenge.

### Connecting to MySQL Server

**PHP** se MySQL server ko connect karne ke liye hum **mysqli_connect()** function ka use karte hain. Is function ke andar hume 4 cheezein deni hoti hain:

1. **Hostname**: Jahaan par MySQL server run ho raha hai. (Localhost agar aapka server apne machine pe hai)
2. **Username**: MySQL server ka username (usually **root** by default).
3. **Password**: MySQL server ka password (agar set kiya ho).
4. **Database name** (optional, but needed for selecting a database).

**Code**:

```
<?php
  // MySQL server se connect ho rahe hain
  $conn = mysqli_connect("localhost", "root", "", "my_database");

  // Connection check kar rahe hain
  if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
  }

  echo "Connected successfully";
?>
```

**Explanation**:

- `mysqli_connect("localhost", "root", "", "my_database")`: Yahan pe localhost ka matlab hai ki MySQL server apne hi machine par run ho raha hai. Agar remote server pe hai toh uska IP ya domain name dena padta.
- Agar connection nahi hota, to `mysqli_connect_error()` function error message deta hai.

---

## 2. Selecting Databases

MySQL server pe jab aap multiple databases use karte hain, tab aapko specific database select karna padta hai jisme aapka kaam ho raha ho. **`mysqli_select_db()`** function ka use karke aap selected database ko choose karte hain.

**Code**:

```
<?php
 // Connection establish karne ke baad
 $conn = mysqli_connect("localhost", "root", "");

 // Agar connection fail ho jaye
 if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
 }

 // Database select kar rahe hain
 $db_select = mysqli_select_db($conn, "my_database");

 // Agar database select nahi ho pata
 if (!$db_select) {
   die("Database selection failed: " . mysqli_error($conn));
 }

 echo "Database selected successfully";
?>
```

**Explanation**:

- mysqli_select_db($conn, "my_database"): Yeh function MySQL connection object aur database name ko pass karta hai. Agar database nahi milta, to error message show hota hai.
- mysqli_error($conn): Agar koi error aata hai, to yeh function specific error ka message deta hai.

---

## 1. Checking for Errors

PHP mein **MySQL** ke saath kaam karte waqt errors ko handle karna zaroori hota hai. Jab bhi aap **MySQL server** se connect karte hain ya koi **SQL query** execute karte hain, aapko ensure karna padta hai ki koi error nahi aayi ho. Aap **`mysqli_connect_error()`** aur **`mysqli_error()`** functions ka use karke errors ko handle kar sakte hain.

**Connection Errors:**

- Jab aap MySQL server se connect karte hain, agar connection fail hota hai toh **`mysqli_connect_error()`** function error message dega.

**Query Errors:**

- Jab aap SQL query execute karte hain, agar query mein koi issue hota hai (jaise table ya column nahi milna), toh **`mysqli_error()`** function specific error dikhata hai.

---

## Code for Checking Errors:

```
<?php
  // MySQL server se connect kar rahe hain
  $conn = mysqli_connect("localhost", "root", "", "my_database");

  // Agar connection fail ho
  if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
  }

  // Query ko execute kar rahe hain
  $result = mysqli_query($conn, "SELECT * FROM non_existent_table");

  // Agar query fail ho
  if (!$result) {
    die("Query failed: " . mysqli_error($conn));
  }
?>
```

---

## Closing the MySQL Server Connection

Aapka kaam complete hone ke baad MySQL server se connection ko close karna zaroori hota hai, taki server ki resources free ho sakein. Iske liye hum **`mysqli_close()`** function ka use karte hain. Yeh function MySQL connection ko safely band kar deta hai.

## Code for Closing Connection:

```
<?php
  // Connection establish karte hain
  $conn = mysqli_connect("localhost", "root", "", "my_database");

  // Agar connection fail ho
  if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
  }

  // Connection close karte hain
  mysqli_close($conn);
?>
```

## 1. Inserting Records in MySQL Using PHP

MySQL database mein naye records ko insert karne ke liye **`INSERT INTO`** SQL query ka use hota hai. PHP mein is query ko execute karne ke liye **`mysqli_query()`** function use kiya jata hai. Aap **`$_POST`** ya **`$_GET`** superglobals se form ke data ko PHP mein receive karke SQL query mein insert kar sakte hain.

- **INSERT Query**: Iska use records ko ek table mein add karne ke liye hota hai.

**Code for Inserting Data:**

```
INSERT INTO users (name, email) VALUES ('Kundan Bharti', 'kundanbharti@gmail.com');
```

---

## 2. Viewing Records in MySQL Using PHP

Agar aapko MySQL database se records dekhna ho, toh `SELECT` query ka use hota hai. `mysqli_query()` function se query execute karke, aap `mysqli_fetch_assoc()` ya `mysqli_fetch_array()` functions se records ko fetch kar sakte hain aur PHP mein display kar sakte hain.

- **SELECT Query**: Yeh query database se data retrieve karne ke liye use hoti hai.

```
SELECT * FROM users;
```

---

## 3. Updating Records in MySQL Using PHP

**Theory**:
Agar aapko database mein kisi record ko update karna ho, toh **`UPDATE`** query ka use karte hain. **`SET`** keyword ka use karke aap columns ke values ko update karte hain. Aapko **`WHERE`** clause ka use karna padta hai taaki aap specific records ko hi update kar sakein, nahi toh saare records update ho jayenge.

- **UPDATE Query**: Yeh query table ke existing records ko modify karne ke liye use hoti hai.

```
UPDATE users SET name = 'Kundan' WHERE id = 1;
```

---

## Deleting Records in MySQL Using PHP

Agar aapko kisi record ko delete karna ho, toh **`DELETE FROM`** query ka use hota hai. Isme bhi **`WHERE`** clause ka use zaroori hai taaki aap specific records hi delete karein. Agar **`WHERE`** clause nahi diya, toh saare records delete ho sakte hain.

- **DELETE Query**: Yeh query table se records ko remove karne ke liye use hoti hai.

```
DELETE FROM users WHERE id = 1;
```

## Manipulating Joined Tables in MySQL Using PHP

Agar aapko multiple tables se data retrieve karna ho, toh **`JOIN`** query ka use hota hai. **`INNER JOIN`**, **`LEFT JOIN`**, **`RIGHT JOIN`**, aur **`FULL JOIN`** SQL mein available hain. **`JOIN`** ka use aap tab karte hain jab aapko ek se zyada tables ka data ek saath fetch karna ho. Aap **`ON`** keyword ka use karke specify karte hain ki kis column ko join karna hai.

- **JOIN Query**: Yeh query multiple tables ko combine karke data fetch karne ke liye use hoti hai.

```
SELECT users.name, orders.order_date
FROM users
INNER JOIN orders ON users.id = orders.user_id;
```

---

## Creating Session (Session Banane ki Process)

**User Authentication** ka ek important part hai **session management**. Jab bhi koi user apne credentials (jaise username aur password) ke through login karta hai, tab us user ki identity ko **verify** karna aur login ke baad **uski session** ko maintain karna zaroori hota hai. Session ka main kaam hai ki user ko ek unique ID assign karna aur uske saare requests ko track karna, taki har request mein user ki identity confirm ho sake.

- **Session**: Yeh ek temporary storage hota hai, jisme user ki information store hoti hai jaise user ka naam, login time, role, etc.
- **PHP mein Session**: PHP mein session start karne ke liye **`session_start()`** function ka use karte hain. Iske baad aap **`$_SESSION`** superglobal array mein user-specific data store kar sakte hain.

**Session Start and Storing Data**:

- Jab user successfully login kar leta hai, tab hum session start karte hain aur uski details **`$_SESSION`** array mein store karte hain.

**Code Example for Creating Session:**

```
<?php
  session_start(); // Session ko start karte hain

  // User login hone ke baad session mein data store karte hain
  $_SESSION['username'] = "Kundan"; // Username store kar rahe hain
  $_SESSION['role'] = "admin"; // User ka role store kar rahe hain

  echo "Session created successfully!";
?>
```

---

## Authorization Level (Authorization ka Concept)

**Authorization** ka matlab hota hai ki ek authenticated user ko kaunse actions ya resources access karne ki permission di jaye. Jab user successfully authenticate ho jata hai, tab uski authorization level ko set kiya jata hai. Yeh permission uske role ke basis pe hoti hai, jaise **admin**, **user**, **editor**, etc.

- **Authorization** ka purpose hota hai ki kisi bhi user ko restricted areas ya sensitive data tak access na ho. Jaise **admin** ko sabhi pages aur actions tak access hota hai, lekin **normal user** ko sirf limited pages aur actions ki permission hoti hai.
- **Role-Based Access Control (RBAC)** ka concept use hota hai, jisme user ko uske role ke according specific permissions diye jaate hain.

**PHP mein Authorization**:

- Aap user ki role ya authorization level ko **`$_SESSION`** mein store karte hain aur phir uske access level ke hisaab se check karte hain ki user kisi page ya resource ko access kar sakta hai ya nahi.

**Code Example for Authorization Level:**

```
<?php
  session_start(); // Session start karte hain

  // Check karte hain ki user logged in hai ya nahi
  if (isset($_SESSION['username']) && $_SESSION['role'] == "admin") {
    echo "Welcome Admin!";
  } else {
    echo "You are not authorized to access this page!";
  }
?>
```

## Unit-V Web Hosting

### Uploading Web Pages - Using FTP (FTP ka Use karke Web Pages Upload Karna)

**FTP (File Transfer Protocol)** ek protocol hai jisse hum apne local machine se web server par files ko upload ya download kar sakte hain. FTP ka use karte hue, hum apne web pages ko ek web server par upload karte hain taaki wo duniya bhar mein accessible ho sake.

- **FTP Client**: FTP client ek software hota hai jisme aap apne FTP server se connect hote hain aur files ko easily transfer karte hain. Popular FTP clients mein **FileZilla** aur **WinSCP** hain.
- **Steps to Upload Web Pages using FTP**:
    1. FTP client ko install karen.
    2. Server details (hostname, username, password) enter karein.
    3. Local machine se files select karen aur remote server par upload karen.

**Advantages of FTP**:

- Easy way to transfer large files.
- Allows direct access to server files.

---

**Code Example for FTP Upload (PHP mein FTP ka use):**

```
<?php
  $ftp_server = "ftp.example.com";  // FTP server ka address
  $ftp_username = "username";       // FTP username
  $ftp_password = "password";       // FTP password

  // FTP server se connect hona
  $conn_id = ftp_connect($ftp_server);

  // Login hona FTP server par
  if (ftp_login($conn_id, $ftp_username, $ftp_password)) {
    echo "Connection successful!";
  } else {
    echo "Connection failed!";
  }

  // File ko upload karna
  $file = "localfile.html";  // Local file path
  $remote_file = "remote_file.html";  // Server par file ka naam
  ftp_put($conn_id, $remote_file, $file, FTP_ASCII);

  // Connection close karna
  ftp_close($conn_id);
?>
```

## Using Web Page Editors (Web Page Editors ka Use Karna)

**Web Page Editors** woh software hote hain jo humare liye web pages create karne mein help karte hain. Yeh editors graphical interface provide karte hain, jisme coding aur design ka kaam bahut asaan ho jata hai.

- **WYSIWYG Editors (What You See Is What You Get)**: In editors mein aapko jo aap screen par dekhte hain, wahi result aapke web page par dikhai deta hai. Ismein aapko coding ki zaroorat nahi hoti, bas drag-and-drop se design kar sakte hain.
  - **Examples**: **Adobe Dreamweaver**, **Wix**, **WordPress**.
- **Text-Based Editors**: Yeh basic text editors hote hain, jisme aapko manually code likhna padta hai.
  - **Examples**: **Notepad++, Sublime Text**, **Visual Studio Code**.

**Advantages**:

- WYSIWYG editors se design ka process fast hota hai.
- Text-based editors mein control zyada hota hai, lekin coding skills ki zaroorat padti hai.

---

## Web Hosting (Web Hosting ke Types)

**Web Hosting** ka matlab hai apne website ko ek server par store karna taaki wo internet par accessible ho sake. Web hosting ka ek server par storage space milta hai jahan aap apne web pages ko host karte hain.

- **Shared Hosting**: Ismein ek server par multiple websites hosted hoti hain. Sab websites ko ek server ka space milta hai. Yeh budget-friendly hota hai aur small websites ke liye ideal hai.
  - **Advantages**: Cost-effective, Easy to use.
  - **Disadvantages**: Limited resources, Server speed can be affected by other websites on the same server.
- **Dedicated Hosting**: Ismein ek server sirf ek website ke liye dedicated hota hai. Yeh high-performance websites ke liye use hota hai.
  - **Advantages**: High control, better performance.
  - **Disadvantages**: Expensive, Requires more technical knowledge.
- **VPS Hosting (Virtual Private Server)**: Yeh shared hosting aur dedicated hosting ka mixture hota hai. Ismein aapko ek server ka partition milta hai, jisme zyada control aur resources hote hain.
  - **Advantages**: More control, better performance.
  - **Disadvantages**: More expensive than shared hosting.

---

## Running a Local Web Server (Local Web Server Chalana)

**Local Web Server** ka matlab hai apne local machine par ek server setup karna, jisme aap apne web pages ko test kar sakte hain bina internet ki zaroorat ke. Yeh local development ke liye bahut useful hai.

- **Local Server Software**:
  - **XAMPP**: XAMPP ek free and open-source cross-platform web server solution hai, jo Apache, MySQL, PHP aur Perl ko ek saath combine karta hai. Aap apne local machine par **XAMPP** install karke apne web pages ko run kar sakte hain.
  - **WAMP**: WAMP bhi ek similar tool hai jo Windows OS ke liye available hai. Yeh PHP, MySQL aur Apache ko integrate karta hai.
  - **MAMP**: MAMP MacOS ke liye hota hai, jo Apache, MySQL, aur PHP ko local development ke liye provide karta hai.
- **Local Server Setup**:
  1. XAMPP ya WAMP ko install karen.
  2. Apache aur MySQL services ko start karen.
  3. Apne web pages ko **htdocs** folder mein rakhkar browser mein **localhost** ke through access karen.