MODULE 8 JAVA WEB TECHNOLOGIES

HTML Theory

1. **Introduction to HTML and its Structure:**
   HTML (HyperText Markup Language) is the foundational language for creating webpages. It defines the structure and content of web documents. An HTML document starts with a <!DOCTYPE html> declaration, followed by the <html> element which contains <head> and <body>. The <head> holds metadata like title, scripts, and styles, while the <body> contains the visible content. HTML organizes content into elements signified by tags, creating a tree-like structure that browsers render.

2. **Key HTML Tags and Their Purpose:**

   - a. **<a> (Anchor Tag):** Creates hyperlinks that allow navigation to other web pages, files, or sections within a page. It uses the href attribute to specify the URL or location.
   - b. **<form> (Form Tag):** Used to collect user input. Forms contain input controls like text fields, radio buttons, checkboxes, and submit buttons, enabling data submission to servers via methods like GET or POST.
   - c. **<table> (Table Tag):** Represents tabular data with rows (<tr>) and cells (<td> for data, <th> for headers). Used for structured data presentation.
   - d. **<img> (Image Tag):** Embeds images into the webpage. The src attribute specifies the image source, while alt text improves accessibility for users who cannot see images.
   - e. **List Tags:**
     - <ul> creates an unordered (bulleted) list.
     - <ol> creates an ordered (numbered) list.
     - <li> defines individual list items within both <ul> and <ol>.
   - f. **<p> (Paragraph Tag):** Defines paragraphs or blocks of text, separating content into readable chunks.
   - g. **<br> (Line Break):** Inserts a single line break, useful inside paragraphs or text blocks to control spacing.
   - h. **<label> (Label Tag):** Associates text labels with form elements to improve usability and accessibility, enabling users to click on labels to focus inputs.

CSS Theory

3. **Overview and Importance of CSS in Web Design:**
   CSS (Cascading Style Sheets) is used alongside HTML to dictate the visual presentation of webpages, controlling styles like colors, fonts, spacing, alignment, and responsive layouts. Separating content (HTML) from style (CSS) greatly enhances maintainability, consistency across pages, and user experience by enabling flexible designs adapting to different devices.

4. **Types of CSS:**

   - a. **Inline CSS:** Applied directly to an HTML element using the style attribute, it affects only that element, often used for quick, specific changes but not recommended for large-scale styling.
   - b. **Internal CSS:** Defined within a <style> tag inside the HTML document's <head>. It applies styles only to that page, useful for single-page customizations.
   - c. **External CSS:** Stored in separate .css files linked via <link> in the <head>, allowing multiple pages to share common styles, facilitating centralized control and improved performance through caching.

5. **Margin and Padding – Definitions and Differences:**
   Margin is the outer space around an element's border, separating it from neighboring elements. Padding is the inner space between an element's content and its border, increasing whitespace inside the box. Understanding their difference is crucial for controlling layout and ensuring content does not touch borders or adjacent elements.

6. **Effect of Margin and Padding on Layout:**
   Margins influence the distance between elements, helping prevent cramped designs and managing flow. Padding improves content readability by adding spacing inside elements, which ensures text or images have room to breathe, enhancing aesthetics and usability without changing external element positioning.

7. **Introduction to CSS Pseudo-Classes:**
   Pseudo-classes select elements based on their state or user interactions rather than document structure. Common pseudo-classes include :hover (when an element is hovered by the cursor), :focus (when elements like input fields are active), and :active (while an element is being clicked). They enhance interactivity and provide visual feedback, making user interfaces intuitive.

8. **Use of Pseudo-Classes:**
   By applying styles dynamically, pseudo-classes improve accessibility and user experience. For example, changing link colors on hover guides users, while highlighting input fields on focus clarifies where users are typing, reducing input errors.

9. **Difference Between ID and Class Selectors in CSS:**
   An ID selector uniquely identifies a single element using the id attribute with # prefix, ensuring styles only apply to one specific element. Class selectors, denoted by ., can be applied to multiple elements, enabling reusable style groups. Best practice is to use IDs for unique page elements (e.g., header, footer) and classes for repeated styles (e.g., buttons, menus).

10. **Usage Scenarios for ID vs Class:**
    Use an ID when styling a unique element that appears once on a page (like the main heading or a special section). Use classes for styling multiple elements sharing styles, such as navigation links, input fields, or repeated content blocks, to keep CSS organized and scalable.

Client-Server Architecture (Java Web)

11. **Overview of Client-Server Architecture:**
    Client-server architecture divides responsibilities where clients (typically browsers) send requests for services, and servers process these requests, perform operations like database queries or calculations, then send responses. This model supports distributing workloads, centralized data control, and scalability in web applications.

12. **Difference Between Client-Side and Server-Side Processing:**
    Client-side processing occurs in the user's browser using languages like JavaScript to handle UI changes instantly without contacting the server. Server-side processing happens on the server, handling core application logic, data storage, authentication, and generating dynamic content sent to clients.

13. **Roles of Client, Server, and Protocols:**
    Clients initiate interactions by sending requests via URLs or forms, servers listen for these requests and provide the appropriate responses. Communication protocols such as HTTP define the rules for message formatting, transmission, and error handling, ensuring reliable and standardized web data exchanges.

HTTP Protocol

14. **Introduction to HTTP and Its Role:**
    HTTP is an application-layer protocol that governs how data is requested and transferred across the web. It orchestrates how browsers interact with servers, sending HTTP requests for resources and receiving responses, including status codes indicating success or errors.

15. **HTTP Request and Response Headers Explanation:**
    Request headers convey information like browser type, accepted formats, cookies, and authentication tokens to help servers respond appropriately. Response headers define content type, caching policies, and other metadata to inform clients about response handling and content attributes.

J2EE Architecture

16. **Introduction to J2EE and Multi-Tier Architecture:**
    J2EE (Java 2 Enterprise Edition) supports enterprise applications using a layered architecture: the presentation layer (UI with Servlets/JSP), business logic layer (EJBs or managed beans handling core processing), and data layer (databases and persistence). This separation promotes modularity, reusability, and scalability.

17. **Role of Containers and Servers:**
    Web containers manage servlet lifecycle and client interactions. Application servers provide enterprise services like transaction management and security for business logic components. Database servers handle persistent storage, querying, and data integrity, enabling fault-tolerant enterprise workflows.

CGI Programming in Java

18. **Introduction to CGI:**
    CGI is an early standard for executing external programs on a web server to create dynamic content. When a client submits data, the server runs the CGI program, which processes input and sends back output as part of the HTTP response.

19. **Process, Advantages, and Disadvantages of CGI:**
    CGI works by spawning a new process per request, passing input and returning output. While simple and widely supported, CGI is inefficient for large-scale use due to high overhead from process creation. It lacks persistence and higher-level abstractions found in modern frameworks like servlets.

Servlet Programming

20. **Introduction to Servlets and How They Work:**
    Servlets are server-side Java programs that handle client requests and generate dynamic web content, running inside servlet containers. Unlike CGI, servlets run in a multithreaded environment, managing multiple requests efficiently without creating new processes.

21. **Advantages and Disadvantages Compared to Other Technologies:**
    Servlets are portable, scalable, and faster than CGI. They offer robust integration with Java APIs and frameworks. However, servlet programming can be complex for beginners, and managing servlet lifecycle and concurrency requires careful coding.

Servlet Versions

22. **History of Servlet Versions:**
    Servlet API has evolved from basic CGI replacements to feature-rich APIs supporting annotations, filters, asynchronous processing, and better session management, aligning with modern web application needs.

23. **Types of Servlets:**

- **GenericServlet:** Protocol-agnostic servlet base class, requiring explicit request/response handling.
- **HttpServlet:** Derived class specialized for HTTP protocol providing overridden methods like doGet() and doPost() for handling HTTP verbs more intuitively.

HttpServlet vs GenericServlet

24. HttpServlet provides HTTP-specific methods simplifying handling of GET, POST, and other HTTP requests, while GenericServlet offers a more generic framework that requires manual protocol handling, making it less commonly used for web apps.

Servlet Life Cycle

25. Servlets go through three primary phases:

- init() initializes servlet resources upon loading,
- service() handles client requests and generates responses repeatedly,
- destroy() performs cleanup before servlet removal, ensuring efficient resource management.

Creating Servlets and Servlet Entry in web.xml

26. Servlets are Java classes configured in web.xml which maps URL patterns to servlets and defines initialization parameters. This declarative approach ensures deployment flexibility and modular configuration separate from code.

Logical URL and ServletConfig Interface

27. Logical URLs abstract complex servlet paths into user-friendly, readable URLs which clients use to access web functionality easily.
28. ServletConfig provides parameters defined in deployment descriptors to servlets for initialization, enabling configuration values to be changed without recompilation.

RequestDispatcher Interface

29. RequestDispatcher facilitates request forwarding or including content from other resources (servlets, JSPs) within the same context. This supports modular design and request reuse.

ServletContext and Web Application Listener

30. ServletContext is a shared object used to store application-wide data accessible by different servlets, supporting inter-component communication and shared resource management.
31. Web application listeners monitor lifecycle events (context creation and destruction), allowing developers to execute code on app startup, shutdown, or session changes (resource management, logging).

Java Filters

32. Filters intercept requests/responses before reaching servlets, used for logging, authentication, encryption, and input validation without changing servlet code.
33. Filters have a lifecycle: they are initialized once, execute across all applicable requests, then destroyed on application shutdown. They are declared and mapped in web.xml.

JSP Basics

34. JSP combines HTML and Java to create dynamic pages. It uses JSTL (JSP Standard Tag Library) for common functionalities, custom tags for reusable components, scriptlets to embed Java code blocks (discouraged in modern apps), and implicit objects (like request, session) that simplify accessing data