AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

ONE SHOT Revision

Unit-1: Introduction

HTML, XML

By Amol sir

Web Technology

Syllabus:

Unit I:

Introduction:

Introduction and Web Development Strategies, History of Web and Internet, Protocols Governing Web, Writing Web Projects, Connecting to Internet, Introduction to Internet services and tools, Introduction to client-server computing.

Web Page Designing:

HTML:

List, Table, Images, Frames, forms.

XML:

Document type definition (DTD), XML schemes, Object Models, presenting and using XML, Using XML, Processors: DOM and SAX.

AKTU PYQs

Introduction

1. Discuss the history of WWW in brief. (AKTU 2022-23)
2. Explain the concept of client and server. Also describe how a Web server is different from normal server. (AKTU 2022-23)
3. Differentiate between Internet and WWW. (AKTU 2022-23)
4. Compare Internet services with protocols. (AKTU 2022-23)
5. Differentiate between HTTP and HTTPS. (AKTU 2022-23)
6. Identify the Dissimilarities between web server and Application server. (AKTU 2021-22)
7. Define web project. (AKTU 2020-21)
8. Explain HTTP get request. (AKTU 2020-21)
9. Define webpage with its type. Discuss responsive webpage with example. (AKTU 2019-20)
10. State the various DNS and protocols used. (AKTU 2019-20)
11. What are the two major protocols for accessing email from servers? (AKTU 2018-19)
12. Explain the HTTP protocol. Mention three features of HTTP that make HTTP a simple but powerful protocol. Give its architecture. (AKTU 2018-19)

AKTU PYQs

HTML

1. Describe <form> tag and its attributes. Also differentiate GET and POST methods. (AKTU 2022-23)
2. Describe frames in HTML. Write markup to create a web page that contains the three vertical columns, having background color as red, blue, green respectively. (AKTU 2022-23)
3. Differentiate between HTML and XML. (AKTU 2021-22)
4. Create an HTML code to create a web page that contains the user registration form with following details user name, user date of birth, user address, user gender, user email id, user mobile number. (AKTU 2021-22)
5. Explain HTML. Compare between HTML and XML. (AKTU 2020-21)
6. Explain the following HTML tags and their attributes: (i) List (ii) Table (AKTU 2020-21)
7. When is it appropriate to use frames? (AKTU 2018-19)
8. What is the use of alternative text in image mapping? (AKTU 2018-19)
9. Create an HTML page named as "Table.html" to display your class time table. (AKTU 2018-19)
   i) Provide the title as Time Table.
   ii) Provide various color options to the cells (Highlight the lab hours and elective hours with different colors.)
10. Discuss how frames play a big role in advertising on web. What roles do form play in making web page dynamic. (AKTU 2018-19)

AKTU PYQs

XML

1. Discuss the applications of XML. Also compare XSD and DTD. (AKTU 2022-23)
2. Design a self-describing XML DTD for storing email data. (AKTU 2022-23)
3. Differentiate between HTML and XML. (AKTU 2021-22)
4. Design a XML DTD for self describing weather report having following details: Date, location, temperature range (Location describes city, state and its country. Country code is unique and not left blank. Temperature range describes high and low temp. in Fahrenheit or Celsius) (AKTU 2021-22)
5. Explain HTML. Compare between HTML and XML. (AKTU 2020-21)
6. Discuss XML. Which technologies are used to define the structure of XML document? Explain and demonstrate with an example. (AKTU 2019-20)
7. Discuss DTD. How the DTD is different from XSD? Demonstrate to create an XML document of 10 students of third year. Add their roll numbers, marks obtained in five subjects, total marks and percentage and validate using DTD. (AKTU 2019-20)
8. What is XML? Create a XML document of 10 students of final CSE. Add their roll numbers, marks obtained in 5 subjects, total marks and percentage. Save this XML document at the server, write a program that accepts student's roll number as input and returns the students marks, total percentage by taking student information for XML document. (AKTU 2018-19)
9. What are XML Parsers? Explain the types of parsers with their advantages and disadvantages. (AKTU 2018-19)

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-1

Today's Target

- AKTU Syllabus Overview
- Introduction
- Web Terminology
- Web Development Strategies

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

Web Technology

"Web technology involves the tools and techniques used to create, manage, and interact with websites and web applications."

- It includes everything from the languages we use to build web pages, to the more complex server-side programming and to interact with databases.

Syllabus:

Overview:

- Unit I: Introduction to Web Technologies, Web Page Designing: HTML and XML.
- Unit II: CSS for styling and designing web pages.
- Unit III: JavaScript for scripting and Networking in Java
- Unit IV: Enterprise Java Beans and Node.js with MongoDB.
- Unit V: Java Servlets and Java Server Pages (JSP).

Web Technology

Learning Outcome(s):

- By the end of this course, you'll be able to build and style web pages, create interactive features, handle server-side programming, and manage interaction with databases.

Career Opportunities:

- Understanding web technologies opens up numerous career opportunities. You could become a front-end designer, back-end developer, or even a full-stack developer.
- Each role is crucial in building and maintaining the websites and applications we rely on.

Web Technology

Introduction

Web:

"The Web or World Wide Web (WWW) a system of interlinked, hypertext documents and resources that are accessed over the Internet."

Difference between 'Web' and 'Internet':

- *They are not the same.*
- The Internet is the global network of interconnected computers/ devices, while the Web is a service that operates over the Internet, allowing users to access documents and applications."

Terminology:

- Webpage
- Website , Web Application and Web Portal
- Web Browser

Introduction

Webpage:

"A webpage is a single document or file that is part of a larger collection on the Web."

- It may contain plain text, hyperlinks, images, audio, video, and animations, and is accessed using a web browser.
- A webpage is identified by its unique *URL (Uniform Resource Locator)*.

Example: A single article on a news website like https://www.webnews.com/news-article

Website:

"A website is a collection of related webpages that are hosted under a single domain name."

- All the pages on a website are interconnected and typically serve a common purpose or belong to the same organization or individual.

Example: https://www.webnews.com is the main domain, and all pages under this URL make up the website.

Introduction

Web Application:

"A web application is a more interactive and dynamic form of a website, where users can perform specific tasks or interact with the content in real-time."

- Unlike static websites, web applications often involve data processing and user input, and they may require login or authentication.

Example: Gmail or Google Docs, where users can send emails or create and edit documents in real-time.

Web Browser:

"A web browser is a software application used to access and display webpages."

- It interprets HTML, CSS, JavaScript, and other web technologies to render content on a user's device.

Example: Chrome, Firefox, and Edge.

Web Development Strategies

- User-Centered Design: Prioritize user experience.
- Responsive Web Design: Adapt to various screen sizes and devices.
- Cross-Browser Compatibility: Consistent across different web browsers.
- Mobile-First Development: Design with mobile devices as the primary focus.
- Performance Optimization: Focus on improving load times and overall site speed.
- Content Management: Create and organize content effectively.
- Search Engine Optimization (SEO): Improve visibility and ranking in search results.
- Security Measures: Protect data and ensure privacy.
- Scalability: Grow with user demand.
- Analytics and Monitoring: Track performance & usage and continuously improve.

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-2

Today's Target

- History of Web
- History of Internet
- Protocols governing Web
- AKTU PYQs

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

History of Web

Sir "Tim Berners-Lee", a British computer scientist born in London, invented the World Wide Web.

Problem: In the late 1980s, while working at CERN, he noticed the challenge of sharing information across different computers and platforms.

Inspiration: Berners-Lee envisioned a solution using hypertext, a technology that was emerging at the time, allowing linked information to be easily shared over the growing internet.

Initial Proposal: In March 1989, he wrote a proposal titled "*Information Management: A Proposal*", which laid the groundwork for the web.

Core Technologies (1990):

HTML (HyperText Markup Language):	Language for structuring web pages.
URI (Uniform Resource Identifier):	Unique address for resources on the web (often called a URL).
HTTP (Hypertext Transfer Protocol):	Protocol for retrieving linked resources from across the web.

History of Web

First Web Tools: Developed the first web browser/editor ("WorldWideWeb.app") and web server ("httpd").

Public Launch: In 1991, the web became available to users outside CERN.

Open Web Philosophy: Berners-Lee ensured the web was free to use, without proprietary control, to encourage widespread adoption.

W3C (World Wide Web Consortium): Founded in 1994 by Berners-Lee to develop open web standards.

History of Internet

Early Development (1960s): The concept of the internet began in the 1960s with the development of packet switching, a method of data transmission, by researchers like Paul Baran and Donald Davies.

ARPANET (1969): The Advanced Research Projects Agency Network (ARPANET), funded by the U.S. Department of Defense, became the first operational packet-switching network, connecting universities and research institutions.

TCP/IP Protocol (1970s): In the 1970s, Vint Cerf and Bob Kahn developed Transmission Control Protocol/Internet Protocol (TCP/IP), the foundational protocol suite that enabled different networks to communicate with each other, forming the internet.

First Email (1971): The first email was sent by Ray Tomlinson, a computer engineer, who also introduced the "@" symbol for email addresses.

History of Internet

Networking Expansion (1980s): Throughout the 1980s, ARPANET expanded and connected to other networks, forming a large, decentralized system. In 1983, TCP/IP became the standard for ARPANET, marking the birth of the modern internet.

Commercialization (1990s): In the early 1990s, restrictions on the commercial use of the internet were lifted, leading to the rapid growth of internet service providers (ISPs) and widespread adoption.

Birth of the World Wide Web (1991): Tim Berners-Lee created the World Wide Web, a system of interlinked hypertext documents, making the internet more accessible to the general public.

Protocols governing Web

"A protocol is a set of rules or standards that define how data is transmitted and received over a network."

- It ensures that devices (like computers, phones, servers) can communicate with each other effectively, regardless of differences in hardware or software.
- In simple terms, it's like a common language that devices use to talk to each other over the network.

Hypertext Transfer Protocol (HTTP):

"HTTP is a protocol used for transferring data over the web."

- It defines the rules for communication between a web client (usually a browser) and a web server.
- HTTP is the foundation of any data exchange on the Web, enabling the fetching of resources, such as HTML documents, images, and other web content.

Protocols governing Web

Features of HTTP:

Request/Response Model:

HTTP follows a simple request-response cycle, where the client sends a request and the server responds with the data.

Statelessness:

Each HTTP request is independent and does not rely on previous requests. This simplifies the protocol because the server does not need to retain any session information between requests, allowing for easy scaling of web applications.

Media Independence:

It means, any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content. It is required for the client as well as the server to specify the content type.

Protocols governing Web

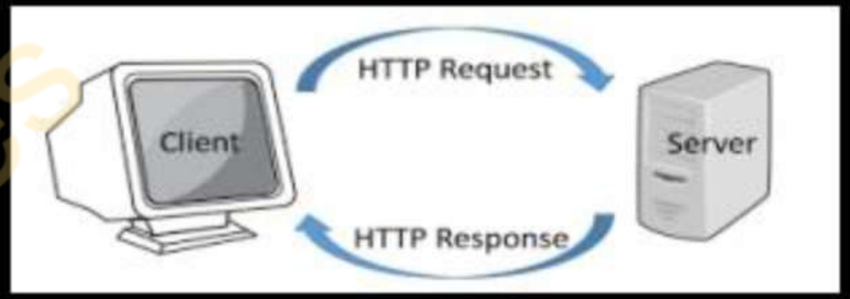
Architecture of HTTP:

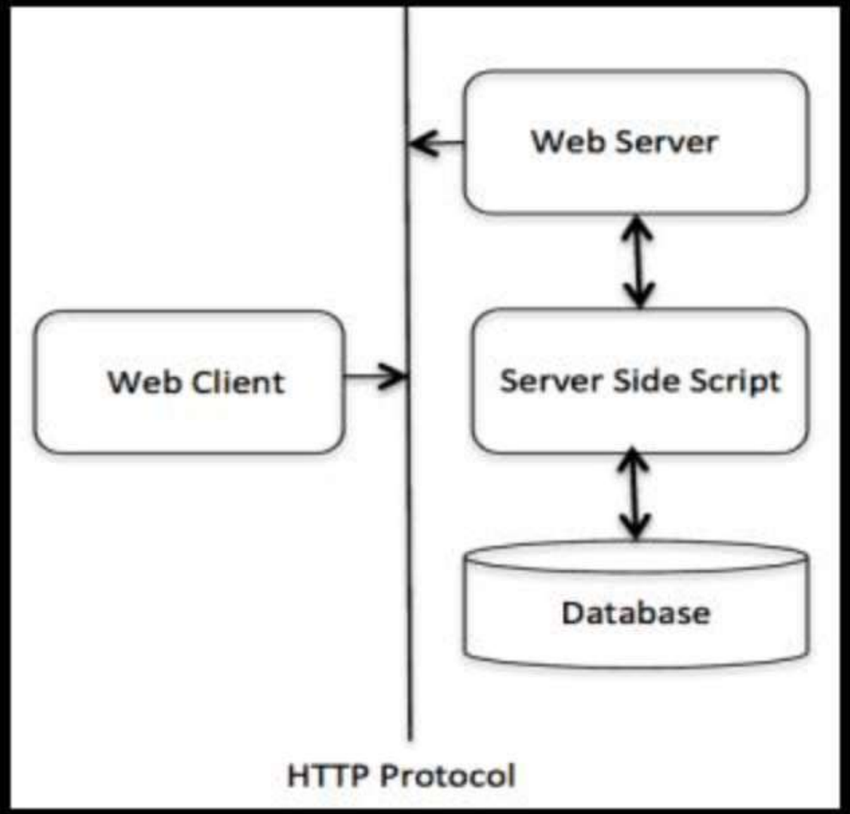
HTTP Request

After establishing a TCP connection to the web server, the browser sends requests, known as HTTP requests, to that web server. An HTTP request specifies the file the browser wishes to receive from the web server along with some additional information, if required.

HTTP Response

When a web server receives an HTTP request, it processes the request and delivers the appropriate content to the browser along with a response header. This response header includes information about the server and the data being returned, including its size and type (such as text or image).





Protocols governing Web

Lack of Confidentiality in HTTP

- The standard HTTP protocol does not provide any means of encrypting its data.
- Because of this lack of encryption, if an attacker could intercept the packets being sent between a web site and a wen browser, he would gain full access to any information the user was transmitting, and could also modify it, as in a *MAN-IN-THE-MIDDLE* scenario.
- This lack of confidentiality therefore makes HTTP inappropriate for the transmission of sensitive information, such as passwords, credit card numbers, and banking transaction data.

Protocols governing Web

Hypertext Transfer Protocol Secure (HTTPS)

- To solve the confidentiality problem inherent in HTTP, an alternative protocol is available called HTTPS.
- HTTPS is identical to HTTP syntactically, but incorporates an additional layer of security known as SSL (secure socket layer), or a newer implementation, known as TLS (transport layer security).
- SSL and TLS rely on the notion of a certificate to verify the identity of the server and establish an encrypted communication channel between the web browser and the web server.

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-3

Today's Target

- Protocols governing Web
  - HTTP, HTTPS, TCP/ IP, SMTP, POP3, IMAP, DNS
- Writing Web Projects,
- Connecting to Internet,
- Introduction to Internet services and tools,
- Introduction to client-server computing.
- AKTU PYQs

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

Protocols governing Web

Transmission Control Protocol/Internet Protocol (TCP/ IP) :

TCP/IP is a set of protocols that allow computers/ devices to communicate with each other on the internet. TCP and IP are the two components of this set of protocols.

1. Transmission Control Protocol (TCP):

TCP establishes a reliable connection between two devices before data is sent, and ensures that the data reaches the other side in the correct order, without any missing parts.

- TCP is like a phone call between two people where both sides ensure they hear every word clearly.
- Imagine sending a set of important documents through a courier who delivers them one by one, but also waits for a confirmation that each one has been received before delivering the next.
- If a document gets lost, TCP ensures it's re-sent until everything is correctly delivered.

Protocols governing Web

2. Internet Protocol (IP):

IP ensures that every data packet traveling over the network has an address and knows where it needs to go.

- Every device on a network has a unique address, called an IP address.
- When data is sent over the internet, IP ensures that the data gets routed correctly from the sender's address to the recipient's address.
- However, unlike TCP, IP doesn't guarantee that the data will arrive safely; it's more like a basic postal system that drops off packets without ensuring that they have been received.

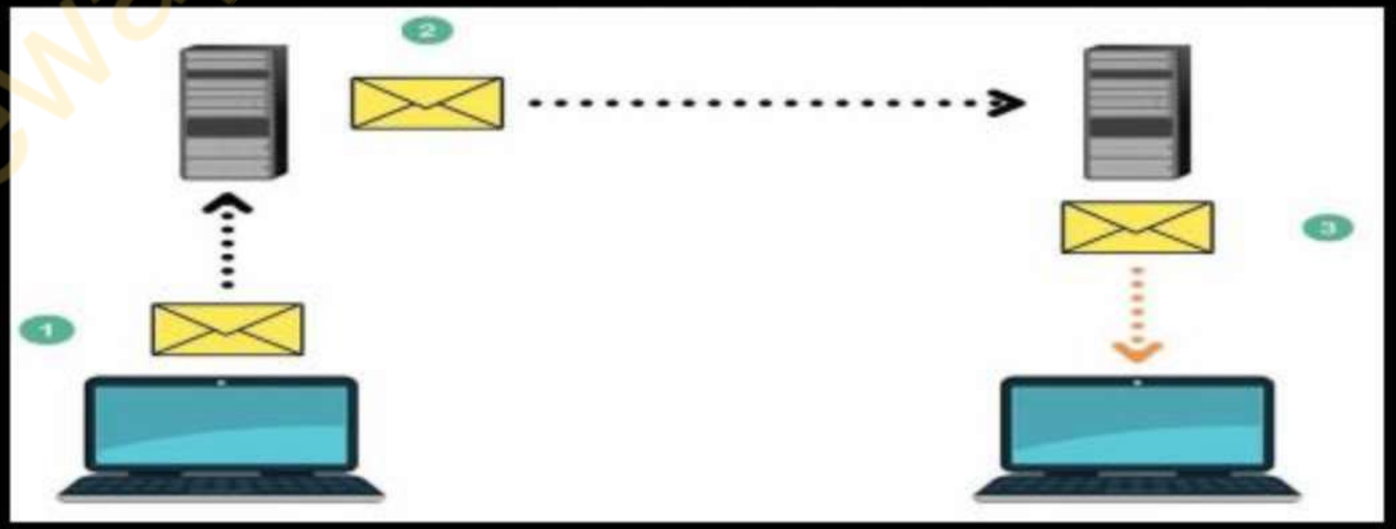
Protocols governing Web

Email Protocols:

- The different email protocols define rules and standards that enable the sending and receiving emails over the Internet.
- They ensure smooth communication between email clients and servers, allowing emails to be properly delivered, retrieved, and managed.

Email Transmission:

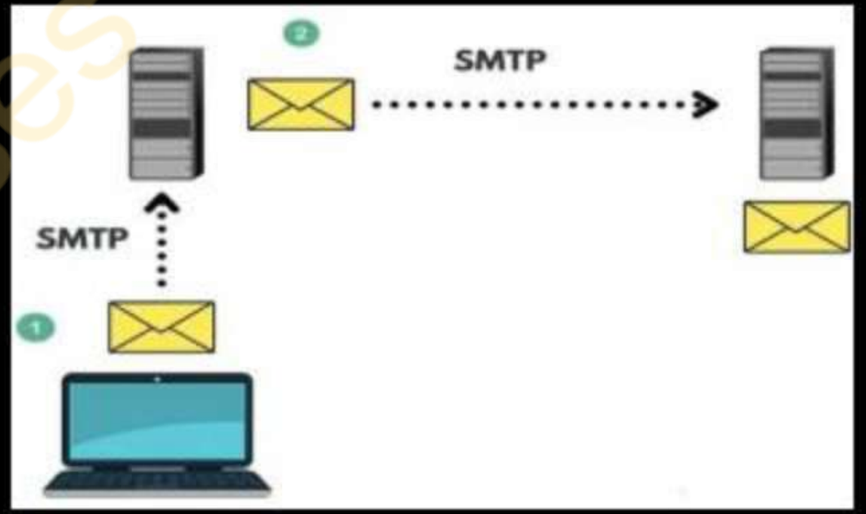
- Sender
- Recipient
- Mail Server(s)



Protocols governing Web

1. SMTP (Simple Mail Transfer Protocol)

SMTP is the protocol used to send emails from an email client to a server and then to the recipient's email server. SMTP is only responsible for sending emails, not receiving or retrieving them.



2. POP3 (Post Office Protocol 3):

POP3 is used to retrieve emails from a server to a single device. Once downloaded, the email is typically removed from the server, so it only exists on that device.

3. IMAP (Internet Message Access Protocol):

IMAP allows you to access and manage emails from multiple devices by keeping the emails stored on the server and synchronizing them across all devices.

Protocols governing Web

DNS (Domain Name System)

This protocol is like an internet phone book that translates user-friendly domain names (like www.example.com) into IP addresses (like 192.0.2.1) that computers use to identify each other on the network.

- When you type a website address into your browser, DNS servers act like directory assistants, finding the correct IP address for that domain and directing your browser to the right location.
- This system makes it easier for people to use memorable domain names rather than having to remember complex numerical IP addresses.

Protocols governing Web

DNS (Domain Name System)

This protocol is like an internet phone book that translates user-friendly domain names (like www.example.com) into IP addresses (like 192.0.2.1) that computers use to identify each other on the network.

- When you type a website address into your browser, DNS servers act like directory assistants, finding the correct IP address for that domain and directing your browser to the right location.
- This system makes it easier for people to use memorable domain names rather than having to remember complex numerical IP addresses.

Writing Web Projects

A web project involves development tasks focused on creating a website or web application to fulfill specific goals or needs. It involves planning, designing, building, testing, deploying, and maintaining the web-based solution.

1. Requirements Definition: Identify and document the needs and goals of the project.
2. Planning: Develop a detailed plan including timelines, resources, and project milestones.
3. Design: Create the visual and functional design of the website, including layout, user interface, and user experience.
4. Development: Build the website according to the design specifications using appropriate technologies.
5. Testing: Evaluate the website for bugs, performance issues, and usability problems to ensure it meets the requirements.
6. Deployment: Launch the website on a live server and make it accessible to users.
7. Maintenance: Regularly update and maintain the website to fix issues, add features, and ensure continued functionality.

Connecting to Internet

Wired Internet Connections

Digital Subscriber Line (DSL)

DSL uses existing telephone lines to transmit data, allowing internet and phone services to operate simultaneously.

Cable

Cable internet utilizes twisted pair or coaxial cables to deliver internet service. It generally provides faster speeds compared to DSL.

Fiber Optics

Fiber optics internet employs light signals transmitted through thin strands of glass or plastic cables, offering very high speeds. It provides superior reliability and speed compared to DSL and cable, with performance largely unaffected by distance.

Connecting to Internet

Wireless Internet Connections

Wireless Fidelity (Wi-Fi)

Wi-Fi enables internet access within a localized area by using radio waves from a wireless router connected to a broadband service.

Cellular Data

Cellular data provides internet access through mobile networks (3G, 4G, 5G) offered by carriers. It supports mobile and remote internet use, with speeds and coverage varying based on the network generation and signal strength.

Satellite Connection

Satellite internet transmits and receives data via satellites orbiting the Earth, with a satellite dish at the user's location communicating with the satellite network. It's useful for remote areas where other internet types are unavailable but often suffers from higher latency and potential weather-related interruptions.

Internet Services and Tools

Internet Services

Email Services:	send, receive, and manage emails.
Web Hosting Services:	offer space on a server to host websites and web applications.
Cloud Storage:	provide online storage solutions for files and data.
Social Media Platforms:	facilitate social interaction and content sharing.
Streaming Services:	offer on-demand video and audio content.
Online Banking:	perform banking transactions and manage accounts online.

…Internet Services and Tools

Internet Tools

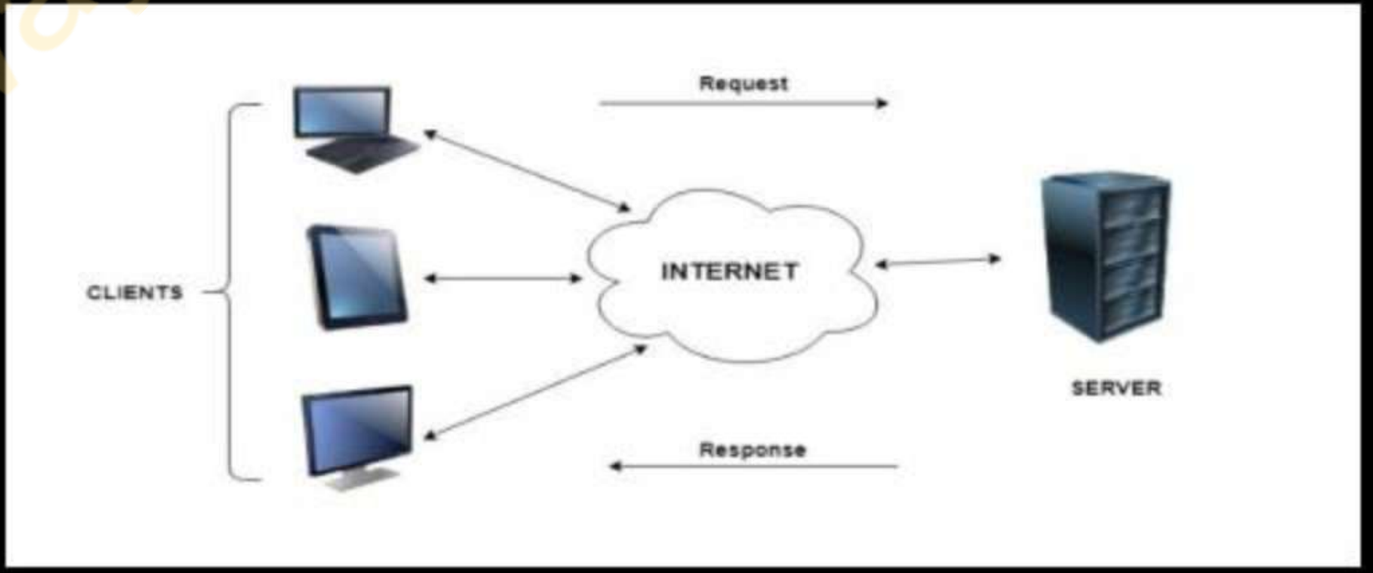
Web Browsers:	to access and navigate websites.
Search Engines:	to find information on the internet.
Productivity Software:	to facilitate document creation, collaboration, and project management.
Communication Tools:	to enable online communication through video calls, messaging, and conferencing.
Development Tools:	to aid in web development and programming.
Security Tools:	to protect users' data and privacy online.

Introduction to Client-Server Computing

In client-server computing, the client requests resources from a server, which then provides the requested resources.

- A server can handle multiple clients simultaneously.
- Although clients and servers typically communicate over a network, they may occasionally reside on the same system.

Example: A web server that delivers web pages to clients who request them.



Characteristics of Client-Server Computing:

- Request-Response Model
- Common Communication Protocol

Introduction to Client-Server Computing

Advantages of Client-Server Computing:

Centralized Data Management: Data is stored in one place (the server), making it easier to protect, authorize, and authenticate.
Geographical Flexibility: The server does not need to be physically close to the clients, yet data access remains efficient.
Independent Nodes: Nodes can be replaced, upgraded, or relocated easily, as they rely on the server for data.
Cross-Platform Compatibility: Clients and servers can operate on different platforms while still facilitating data transfer.

Disadvantages of Client-Server Computing:

Risk of Overloading: A high volume of simultaneous requests from clients can overload the server, causing network congestion.
Single Point of Failure: If the server fails, it can disrupt the entire network, as it cannot fulfill any client requests.
High Costs: Setting up and maintaining a client-server infrastructure can be expensive.

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-4

Today's Target

- Web Page Designing:
  - HTML
    - Document Structure
    - Tags
      - Attributes

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

Hypertext Markup Language (HTML)

"HTML is the standard language used to create and design webpages."

- The "*Hypertext*" part of HTML refers to the capability to link to other documents or resources via hyperlinks, enabling navigation between different pages on the web.
- HTML is called a "*Markup Language*" because it uses tags to "mark up" or define the content.
  - ✓ Tags are enclosed in angle brackets, like <tag name>, and usually come in pairs with an opening tag and a closing tag (e.g., to start a paragraph and to close it).
  - ✓ Tags can define headings, paragraphs, links, images, and more, essentially dictating how content appears in a web browser.
- HTML has evolved over time, with the latest version being HTML5. After this version, it adopted a "*living standard*" model which means HTML is continuously updated with new features and improvements, rather than being released in versions.

Hypertext Markup Language (HTML)

Structure of HTML Document

Document type and version declaration:

<!DOCTYPE html>: Defines the document type and version of HTML.

[Note: The exclamation mark (!) in <!DOCTYPE> indicate that this is a special declaration, not a standard HTML element. It signals to the browser that the following information is about the document type and helps it interpret and render the page correctly.]

HTML document:

<html>..</html>: It defines the beginning and end of the HTML content.

Header section: <head>..</head>: Contains information about web page.

Body section: <body>..</body>: Contains content of web page.

[Note: HTML tags are case-insensitive, meaning you can use uppercase, lowercase, or a mix of both.]

Hypertext Markup Language (HTML)

Attributes of HTML tags:

Attributes in HTML are properties added to HTML tags to provide additional information about the elements.

- *They help define the characteristics or behavior of HTML elements.*

Syntax:

Attributes are always placed within the opening tag of an HTML element, as key-value pairs.

<tagname attribute="value">Content</tagname>

Attributes consist of a name (the key) and a value, separated by an equal sign (=).

Example: <img src="image.jpg" alt="A description of the image">

src: The attribute name (source of the image).

image.jpg: The attribute value (the path to the image file).

Hypertext Markup Language (HTML)

HTML Tags:

<title>: Sets the title of the webpage, displayed in the browser's title bar or tab.

Tags for Text Formatting:

Headings: <h1> to <h6>: Define headings, with <h1> being the highest level.

Line Breaks and Horizontal Rule:

: Inserts a line break. <hr>: Creates a horizontal rule.

[Note:
 and <hr> are empty elements, meaning that these tags do not contain content so they do not have a closing tag.]

Paragraphing: <p>: Defines a paragraph.

Text Styles:

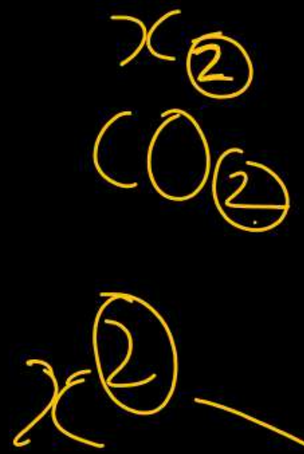
<b>: Bold text. <i>: Italic text. <u>: Underlined text.

Hypertext Markup Language (HTML)

HTML Tags:

Tags for Text Formatting:

Text Styles:



: Important text, usually displayed in bold.

: Emphasized text, usually displayed in italics.

<small>: Smaller text.

<mark>: Highlighted text.

<sub>: Subscript text.

<sup>: Superscript text.

Hypertext Markup Language (HTML)

HTML Tags:

Tags for Text Formatting:

Quotations and Citations: <blockquote>: Long quotations.

<cite>: Cites a reference or work.

Code and Preformatted Text:

<code>: Displays code snippets.

<pre>: Preformatted text, preserving whitespace and line breaks.

Abbreviations and Contact Information:

<abbr>: Abbreviation or acronym, often with a title attribute for full form.

<address>: Contact information for the author or owner of a document.

Hypertext Markup Language (HTML)

HTML Tags:

Tags for Text Formatting:

Text Direction :

<bdi>: Bi-directional isolation for text direction handling.

<bdo>: Overrides the current text direction.

Hyperlinks:

<a>: Defines a hyperlink, with attributes like ***href*** for the destination URL, target for opening links in a new tab, etc.

Hypertext Markup Language (HTML)

HTML Tags:

Tags for Text Formatting:

Text Direction :

<bdi>: Bi-directional isolation for text direction handling.

<bdo>: Overrides the current text direction.

Hyperlinks:

<a>: Defines a hyperlink, with attributes like ***href*** for the destination URL, target for opening links in a new tab, etc.

```
<!DOCTYPE html>
<html>
<head>
	<title> First </title>
</head>

<body>
	My First Web Page

	<h1>Hypertext Markup Language</h1>
	<h2>Hypertext Markup Language</h2>
	<h3>Hypertext Markup Language</h3>
	<h4>Hypertext Markup Language</h4>
	<h5>Hypertext Markup Language</h5>
	<h6>Hypertext Markup Language</h6>

	<p><b>Hypertext Markup language</b> is used to create and
design web pages</p>

	<p><i>Hypertext Markup language</i> is used to create and
design web pages</p>

	<p><u>Hypertext Markup language</u> is used to create and
design web pages</p>

<strong> Web Technology </strong>
<em> Protocols </em>
<small> HTTP </small>
<p> CO<sub>2</sub> is a gas. </p>
<p> x<sup>2</sup> + y<sup>2</sup> </p>
<mark>HTTPS</mark>

<blockquote>Prevention is better than cure</blockquote>
<cite>The good book</cite>

<code>
class Cls
{
	int attrib;

	void method()
	{

	}
}
</code>

<pre>
```

```
class Cls
{
        int attrib;

        void method()
        {

        }
}
</pre>

<abbr title="Hypertext Markup Language">HTML</abbr>

This is something written in arabic <bdi>الرجل القوي إيان</bdi>
<bdo dir="rtl"> Meerut </bdo>
<br>
<hr>
<a href="https://www.google.co.in">Link to Google</a>

</body>
</html>
```

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-5

Today's Target

- Lists in HTML
  - Unordered,
  - Ordered, and
  - Description Lists
- Tables in HTML
  - Key elements of a table

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

Hypertext Markup Language (HTML)

Lists in HTML

"An HTML list is a collection of related items used to display information on web pages. HTML lists can present these items in an ordered sequence, unordered format, or as pairs of terms and descriptions."

There are three main types of lists in HTML5:

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

Hypertext Markup Language (HTML)

Lists in HTML

1. Ordered List (<ol>)

"An ordered list is a numbered list, where each item is numbered in a specific sequence."

It is used when the order of the list items matters, such as in instructions or steps.

Syntax:

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

Hypertext Markup Language (HTML)

Lists in HTML

1. Ordered List (<ol>)

Attributes:

- ***type:*** ***Specifies the type of numbering (e.g., 1, A, a, I, i).***
- ***start:*** ***Specifies the starting number.***

  [Note: The start attribute only accepts numeric values.]

- ***reversed:*** ***Reverses the numbering order.***

Hypertext Markup Language (HTML)

Lists in HTML

2. *Unordered List (<ul>)*

"An unordered list is a bulleted list, where the order of items does not matter."

It is used when the list items can appear in any order, such as a list of features or points.

Syntax:

```
<ul>
  <li>Item one</li>
  <li>Item two</li>
  <li>Item three</li>
</ul>
```

Attributes:

type: Although deprecated in HTML5, it was used to define the bullet style (e.g., disc, circle, square).

Hypertext Markup Language (HTML)

Lists in HTML

3. Description List (<dl>)

"A description list is used to create a list of terms and their descriptions."

It is used when you need to define terms or provide a glossary.

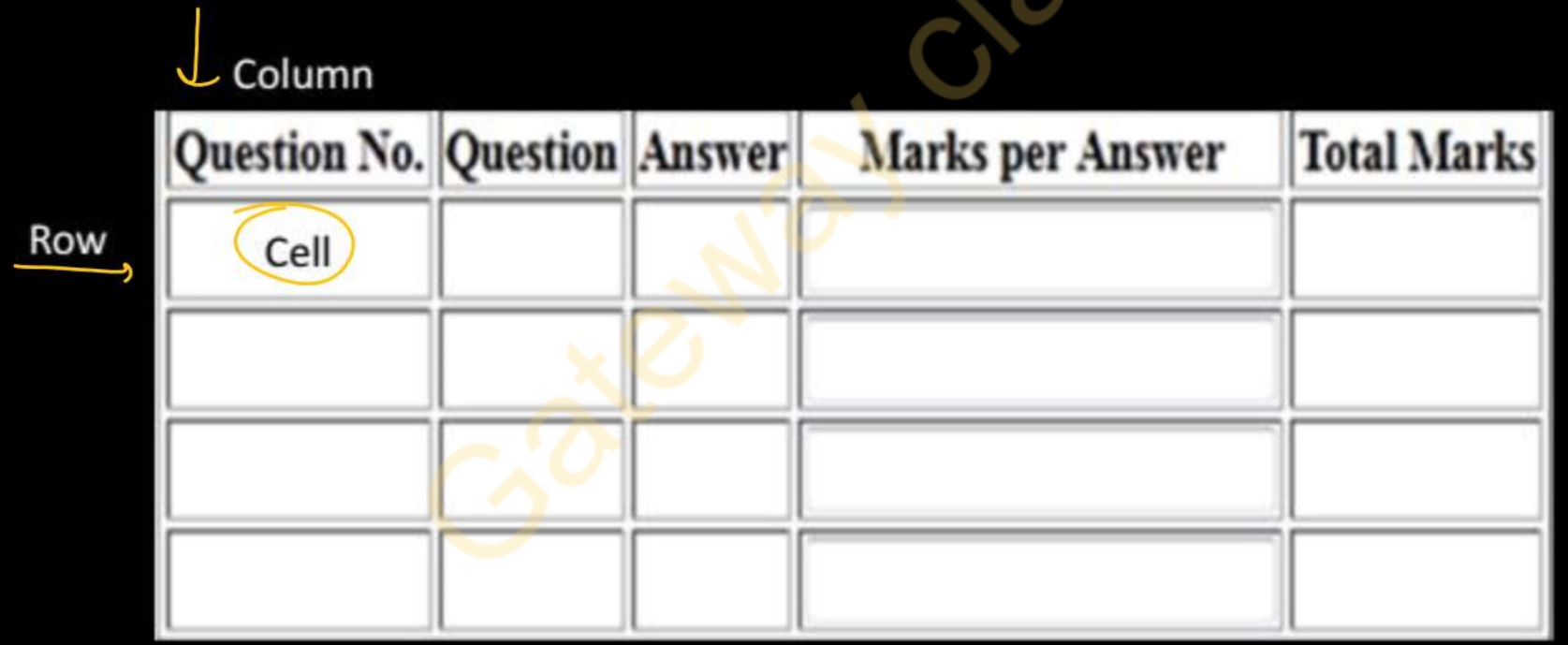
Syntax:

```
<dl>
  <dt>Term 1</dt>
  <dd>Description of Term 1</dd>
  <dt>Term 2</dt>
  <dd>Description of Term 2</dd>
</dl>
```

Hypertext Markup Language (HTML)

Tables in HTML

"Tables in HTML are used to display data in a grid-like format, where the data is organized into rows and columns."



Hypertext Markup Language (HTML)

Tables in HTML

Key Elements of a Table

<table>: This element wraps the entire table.

<caption>: Provides a title or caption for the table. It's optional and usually appears above the table.

Syntax:

```
<table>
<caption>…</caption>
…
</table>
```

Hypertext Markup Language (HTML)

Tables in HTML

Key Elements of a Table

<thead>: Groups the header content in a table. Typically contains the column headings.

<tr> (Table Row): Defines a row in the table.

<th> (Table Header): Defines a header cell in a table. Header cells are bold and centered by default.

<tbody>: *Contains the main body of the table where the data rows are defined.*

<tr>: Defines each row within the <tbody>.

<td> (Table Data): Defines a standard cell in the table where data is stored.

<tfoot>: Contains the footer of the table, often used for summary or totals.

Hypertext Markup Language (HTML)

Tables in HTML

Key Elements of a Table

Syntax:

```
<table>
<thead>
    <tr>
        ( <th>…</th>
        ( <th>…</th>
        ( <th>…</th>
    </tr>
</thead>
```

Hypertext Markup Language (HTML)

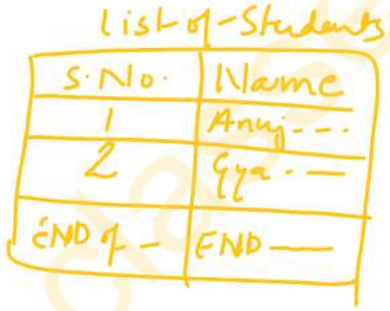
Tables in HTML

Key Elements of a Table

```
<tbody>
    <tr>
        <td>…</td>
        <td>…</td>
        <td>…</td>
    </tr>
</tbody>
<tfoot>
    <tr>
        <td >…</td>
        <td>…</td>
        <td>…</td>
    </tr>
</tfoot>
</table>
```

```
<!DOCTYPE html>
<html>
<head> <title>Lists in HTML</title></head>
<body>
<ol type="1" start="3" reversed>

<li>Design and Analysis of Algorithms</li>
<li>Database Management System</li>
<li>Web Technology</li>

</ol>

<ul type="disc">
<li>HTTP</li>
<li>HTTPS</li>
<li>TCP</li>
</ul>

<dl>
<dt>HTTP</dt>
<dd>HTTP stands for Hypertext Transfer Protocol</dd>

<dt>HTTPS</dt>
<dd>HTTPS stands for Hypertext Transfer Protocol Secure</dd>
</dl>
</body>

</html>
```

```
<!DOCTYPE html>
<html>
<head><title>Tables in HTML</title></head>
<body>

<table border="1">
<caption> List of Students </caption>
<thead>
<tr>
<th> S.NO. </th>
<th> Name </th>
</tr>
</thead>

<tfoot>
<td>END of S.NO.</td>
<td>END of Name</td>
</tfoot>

<tbody>
<tr>
<td>1</td>
<td>Anuj Sharma</td>
</tr>

<tr>
<td>2</td>
<td>Gyanendra Singh</td>
</tr>
</tbody>

</table>

</body>
</html>
```



AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-6

Today's Target

- Tables in HTML
  - Colspan, Rowspan, Cellspacing, and Cellpadding
- Images in HTML
- Frames in HTML
  - Frameset and Frame
- Inline Frames (iframes)
- AKTU PYQs

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

Hypertext Markup Language (HTML)

Tables in HTML

'colspan' and *'rowspan'* attributes

"In HTML tables, 'colspan' and 'rowspan' are attributes used to merge cells across multiple columns or rows, respectively."

colspan: This attribute is used in a table cell (<td> or <th>) to make it span across multiple columns.

```
<table border="1">
 <tr>   <th colspan="3">Header spanning 3 columns</th>  </tr>
 <tr>
  <td>Cell 1</td>   <td>Cell 2</td>   <td>Cell 3</td>
 </tr>
</table>
```

Header spanning 3 columns		
Cell 1	Cell 2	Cell 3

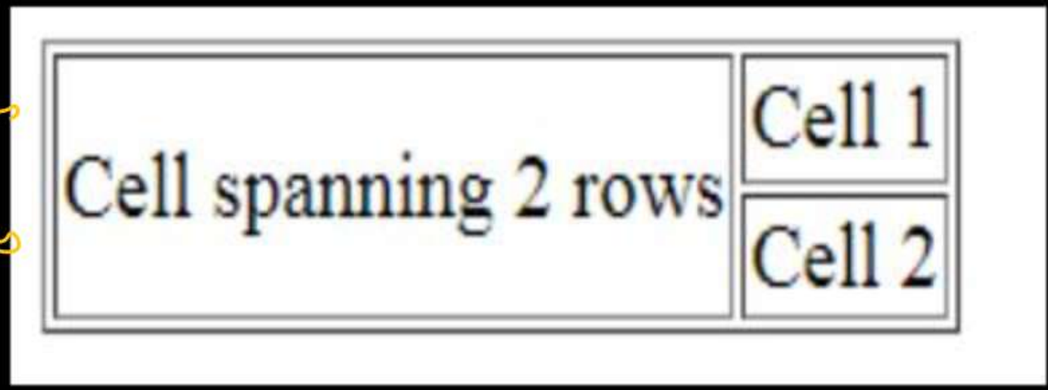
Hypertext Markup Language (HTML)

Tables in HTML

'colspan' and *'rowspan'* attributes

rowspan: This attribute is used in a table cell (<td> or <th>) to make it span across multiple rows.

```
<table border="1">
  <tr>
    <td rowspan="2">Cell spanning 2 rows</td>
    <td>Cell 1</td>
  </tr>
  <tr>
    <td>Cell 2</td>
  </tr>
</table>
```



Hypertext Markup Language (HTML)

Tables in HTML

PYQ: Create an HTML page named as "Table.html" to display your class time table. (AKTU 2018-19)

i) Provide the title as Time Table.

ii) Provide various color options to the cells (Highlight the lab hours and elective hours with different colors.)

Write the HTML code to create the following table as described in the question:

Time Table

CSE			Semester-V		2024-2025		
Day	09:00 - 10:00	10:00 - 11:00	11:00 - 12:00	12:00 - 01:00	01:00 - 02:00	02:00 - 03:00	03:00 - 04:00
MON	BCS-502	BNC-601	BCS-501	LUNCH	BCS-055	BCS-503	BCS-054
TUE	BCS-502	BCS-501	BCS-055	LUNCH	BCS-503	BCS-551	
WED	BCS-503	BCS-054	BCS-502	LUNCH	BCS-501	BCS-552	
THU	BNC-601	BCS-054	BCS-055	LUNCH	BCS-502	BCS-553	
FRI	BCS-054	BCS-503	BCS-501	LUNCH	BCS-055	BCS-554	

S.No.	Code	Subject Name	Faculty Member
1	BCS-501	Database Management Systems	Ms. XXX
2	BCS-502	Web Technology	Mr. XXX
3	BCS-503	Design and Analysis of Algorithm	Ms. XXX
4	BCS-054	Object Oriented System Design with C++	Mr. XXX
5	BCS-055	Machine Learning Techniques	Ms. XXX
6	BCS-551	Database Management Systems Lab	Mr. XXX
7	BCS-552	Web Technology Lab	Ms. XXX
8	BCS-553	Design and Analysis of Algorithm Lab	Mr. XXX
9	BCS-554	Mini Project or Internship Assessment	Ms. XXX
10	BNC-601	Constitution of India, Law and Engineering	Mr. XXX

Time Table

CSE			Semester-V		2024-2025		
Day	**09:00 - 10:00**	**10:00 - 11:00**	**11:00 - 12:00**	**12:00 - 01:00**	**01:00 - 02:00**	**02:00 - 03:00**	**03:00 - 04:00**
MON	BCS-502	BNC-601	BCS-501	LUNCH	BCS-055	BCS-503	BCS-054
TUE	BCS-502	BCS-501	BCS-055	LUNCH	BCS-503	BCS-551	
WED	BCS-503	BCS-054	BCS-502	LUNCH	BCS-501	BCS-552	
THU	BNC-601	BCS-054	BCS-055	LUNCH	BCS-502	BCS-553	
FRI	BCS-054	BCS-503	BCS-501	LUNCH	BCS-055	BCS-554	

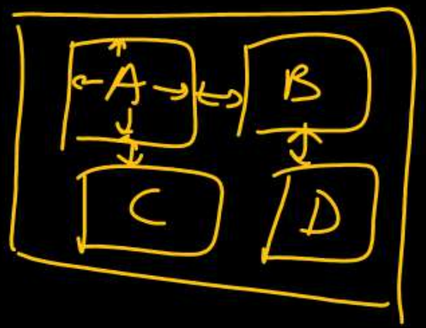
S.No.	Code	Subject Name	Faculty Member
1	BCS-501	Database Management Systems	Ms. XXX
2	BCS-502	Web Technology	Mr. XXX
3	BCS-503	Design and Analysis of Algorithm	Ms. XXX
4	BCS-054	Object Oriented System Design with C++	Mr. XXX
5	BCS-055	Machine Learning Techniques	Ms. XXX
6	BCS-551	Database Management Systems Lab	Mr. XXX
7	BCS-552	Web Technology Lab	Ms. XXX
8	BCS-553	Design and Analysis of Algorithm Lab	Mr. XXX
9	BCS-554	Mini Project or Internship Assessment	Ms. XXX
10	BNC-601	Constitution of India, Law and Engineering	Mr. XXX

Hypertext Markup Language (HTML)

Tables in HTML

'cellspacing' and *'cellpadding'* attributes

'cellspacing' and ***'cellpadding'*** are attributes of the <table> tag used in older versions of HTML to control the spacing in tables:



cellspacing: This attribute sets the space between individual table cells.

For example, cellspacing = "10" adds 10 pixels of space between each cell.

cellpadding: This attribute defines the space between the content of a cell and its border.

For instance, cellpadding = "5" adds 5 pixels of padding inside each cell.

[Note: In modern HTML5, these attributes are deprecated, and it's recommended to use CSS properties like border-spacing (for cellspacing) and padding (for cellpadding) to achieve similar effects.]

Hypertext Markup Language (HTML)

Images in HTML

Images are an essential part of web content and they can improve the design and the appearance of a web page. HTML provides <img> tag to include them in web pages.

Syntax:

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

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The <img> tag has two required attributes:

1. *src (Source) - Specifies the path to the image file. It can be a relative path, absolute path, or a URL.*
2. *alt (Alternative Text) - Specifies an alternate text for the image it it cannot be displayed for any reason.*

Apart from the above, width and height attributes may be used to set the size (dimensions) of the image. We can specify values in pixels or percentages.

Hypertext Markup Language (HTML)

Frames in HTML

"Frames are used to divide a browser window in to multiple sections where each section, or "frame," is an independent window that can load a separate HTML file. A collection of frames in the browser window is known as a frameset."

Frames allow multiple HTML documents to be displayed within a single browser window.

Frame Elements:

<frameset>: Replaces the <body> tag when using frames. It defines how to divide the window into frames. The rows attribute of <frameset> tag defines horizontal frames and cols attribute defines vertical frames.

<frame>: Represents a particular frame and specifies the HTML document that should be loaded into each frame. The src attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. The name attribute gives a name to a frame, allowing links in one frame to target and load pages into another.

<noframes>: Provides content for browsers that do not support frames.

[*Note: Frames are deprecated in HTML5 and should not be used in modern web development. Use the <iframe> tag to embed another document within the current HTML document.*]

Hypertext Markup Language (HTML)

Frames in HTML

Absolute values in pixels

```
<frameset cols="25%, 75%">
    <frame src="menu.html">
    <frame src="content.html">
    <noframes>
        <body><p>The browser you are using does not support frames.</p></body>
    </noframes>
</frameset>
```

Percentage of the browser window

```
<frameset rows="100, 500">
    <frame src="menu.html">
    <frame src="content.html">
</frameset>
```

Using a wildcard symbol

```
<frameset cols="25%, *, 25%">
    <frame src="top.html">
    <frame src="menu.html">
    <frame src="content.html">
</frameset>
```

Nested Framesets:

```
<frameset rows="20%, 80%">
    <frame src="header.html">
    <frameset cols="30%, 70%">
        <frame src="sidebar.html">
        <frame src="main.html">
    </frameset>
</frameset>
```

Hypertext Markup Language (HTML)

Frames in HTML

Advantages:

1. Independent Navigation: Frames allow users to navigate different sections of a site independently, enabling simultaneous viewing of content and menus.
2. Consistent Layout: They provide a way to maintain a consistent layout by keeping certain elements (like navigation) fixed while other content changes.
3. Easy Updates: Advertisements or content in frames can be updated easily without affecting the overall webpage, simplifying maintenance.
   - ➢ Frames play a significant role in web advertising by allowing a fixed section of a webpage to display ads while the rest of the content remains independent and scrollable.
   - ➢ This ensures that ads remain visible even when users navigate different parts of the page.
   - ➢ Frames also allow for easy updating and management of ad content without altering the main webpage, making them a convenient tool for advertisers.
4. Multiple Content Displays: Frames enable the display of multiple documents simultaneously, allowing for richer content interaction.

Hypertext Markup Language (HTML)

Frames in HTML

Disadvantages:

1. Difficult Bookmarking: Users cannot bookmark specific pages within a frameset, making it hard to return to specific content.
2. Not Mobile-Friendly: Some smaller devices cannot cope with frames due to limited screen size, leading to a poor user experience.
3. Screen Resolution Variability: Pages may display differently on different computers because of varying screen resolutions, causing layout inconsistencies.
4. Browser Back Button Issues: The browser's back button may not function as users expect, complicating navigation.
5. Limited Browser Support: A few browsers do not support frame technology, which can result in incomplete functionality for some users.

Hypertext Markup Language (HTML)

Inline Frame (iframe) in HTML

"An iframe (inline frame) allows to embed an HTML document in the other HTML document."

Syntax:

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

The iframe tag allows you to embed content directly within a webpage alongside other elements, providing greater flexibility.

While both frames and iframes serve the purpose of embedding resources, they are fundamentally different:

- frames are primarily used for defining layout, whereas
- iframes focus on adding content.

This focus on content makes iframes advantageous, as they enhance accessibility, improve SEO by allowing better indexing, and offer security features that protect against vulnerabilities. Additionally, iframes can easily adapt to different screen sizes, making them more suitable for modern, responsive web design.

```
<!DOCTYPE html>
<head>
        <title>Tables in HTML</title>
</head>
<body>

<table border="1" cellspacing="10" cellpadding="10">
<tr>
        <th>Name</th>
        <th>Course</th>
        <th>Fees</th>
</tr>
<tr>
        <td colspan="2">ABC</td>
        <td rowspan="2">1000</td>
</tr>
<tr>
        <td>ABC</td>
        <td>Web Technology</td>
</tr>
</table>

<img src="E:/WT_Programs/Unit - I/Images/GatewayLogo.png" width = "500" height = "500" alt="This is
the logo of Gateway Classes">

</body>
</html>
```

FramesDemo.html

```
<!DOCTYPE html>
<html>
<head><title>Frames in HTML</title></head>
<frameset rows="20%, 80%">
        <frame src="one.html">
        <frameset cols="30%, 70%">
                <frame src="two.html">
                <frame src="three.html">
        </frameset>

        <noframes>
                <body><p>Your browser does not support frames!</p></body>
        </noframes>
</frameset>
</html>
```

One.html

```
<!DOCTYPE html>
<html>
        <head><title>Frames in HTML</title></head>

        <body>
                <p> This is one.html file</p>
        </body>
</html>
```

Two.html

```
<!DOCTYPE html>
<html>
        <head><title>Frames in HTML</title></head>

        <body>
                <p> This is two.html file</p>
        </body>
</html>
```

Three.html

```
<!DOCTYPE html>
<html>
        <head><title>Frames in HTML</title></head>

        <body>
                <p> This is three.html file</p>
        </body>
</html>
```

```
<!DOCTYPE html>
<html>
        <head><title>IFrames in HTML</title></head>
        <body>
                <p>Inline Frames in Action</p>
                <iframe src="one.html" width="300" height="200" title="One">
                </iframe>
                <p>I am done with the frames!</p>
        </body>
</html>
```

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-7

Today's Target

- Forms in HTML
  - <form> tag & its attributes
  - GET and POST methods
  - <input> tag
  - Additional Form tags
  - Practice Problems
  - AKTU PYQs

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

Hypertext Markup Language (HTML)

Forms in HTML

"Forms are the medium of interaction between a user and a website or application. Forms allow users to enter data, which is generally sent to a web server for processing."

The <form> Element:

The HTML <form> tag is used to create an HTML form for user input.

```
<form>
        form elements
</form>
```

The <form> is a container for different types of input elements, such as text fields, checkboxes, radio buttons, submit buttons, etc.

Hypertext Markup Language (HTML)

Forms in HTML

Key attributes of <form> tag:

action: *It is used to specify the URL where the form data will be sent after submission.*

For example, <form action="/submit-form.php"> sends the form data to submit-form.php on the server. If omitted, the form submits data to the same page it's on.

method: *It defines the HTTP method used to send the form data. The two common values are GET and POST.*

For example, <form method="POST"> sends data in the request body, while <form method="GET"> appends the data to the URL as query parameters.

Hypertext Markup Language (HTML)

Forms in HTML

abc.com?nm=Anuj&course=BTech

Methods of Submitting Form Data

'GET' method: *The GET method appends form data to the URL in name/value pairs.*

- Sensitive data should never be sent via GET because it will be visible in the URL.
- The GET method is useful for form submissions where the user might want to bookmark the result.
- The length of a URL is limited, typically 2048 characters. This can vary depending on the browser and server configuration.
- GET is more appropriate for non-secure data, such as query strings in search engines like Google.

Hypertext Markup Language (HTML)

Forms in HTML

Methods of Submitting Form Data

'POST' method: *The POST method sends form data in the HTTP request body, not in the URL.*

- Since the data is sent in the body of the request, it is not visible in the URL, making POST more secure for sensitive data.
- The POST method allows for larger amounts of data to be sent, including file uploads and complex data structures.
- Form submissions with POST cannot be bookmarked.

Hypertext Markup Language (HTML)

Forms in HTML

The <input> Element:

The HTML <input> element is the most used form element. An <input> element can be displayed in many ways, depending on the type attribute.

1. **<input type="text">:** Defines a one-line text input field for user input. It can be used for names, addresses, or any other general text.
2. **<input type="password">:** Defines a password field that hides the entered characters.
3. **<input type="submit">:** Defines a submit button that, when clicked, sends the form data to the server specified in the form's action attribute.
4. **<input type="reset">:** Defines a reset button that clears all the input fields in the form to their initial values when clicked.
5. **<input type="radio">:** Defines a radio button that allows the user to select one option from a group. All radio buttons in the same group share the same name attribute.

Hypertext Markup Language (HTML)

Forms in HTML

The <input> Element:

6. **<input type="email">:** Validates that the input is in the correct email format (e.g., user@example.com) and can be used to gather email addresses.
7. **<input type="number">:** Allows the user to enter a numerical value. You can specify min, max, and step attributes to define the range of valid input values.
8. **<input type="checkbox">:** Used for checkboxes, which allow users to select one or more options. Each checkbox can be identified by a unique name attribute.
9. **<input type="date">:** Allows the user to select a date from a calendar widget. The format is typically YYYY-MM-DD.
10. **<input type="time">:** Allows the user to select a time, providing options for hours and minutes. The format is typically HH:MM.

Hypertext Markup Language (HTML)

Forms in HTML

The <input> Element:

11. **<input type="file">:** Allows the user to select one or more files to upload from their device. You can specify the accepted file types using the accept attribute (e.g., accept="image/*" for images).
12. **<input type="search">:** Defines a search field, optimized for search queries. It may have special styling and features in some browsers, such as a clear button.
13. **<input type="tel">:** Defines a telephone number input field. It accepts numeric input and can be used with various input formats depending on the user's locale.
14. **<input type="url">:** Validates that the input is a properly formatted URL (e.g., http://www.example.com).
15. **<input type="color">:** Provides a color picker that allows users to select a color visually. The selected color value is returned in hex format.

Hypertext Markup Language (HTML)

Forms in HTML

The <label> Element:

The <label> tag defines a label for many form elements.

- The <label> element also helps users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.

- The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

Hypertext Markup Language (HTML)

Forms in HTML

The <label> Element:

The <label> tag defines a label for many form elements.

- The <label> element also helps users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.

- The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

Hypertext Markup Language (HTML)

Forms in HTML

The <textarea> Element:

Provides a multi-line text input field for larger amounts of text, such as comments or descriptions. You can specify attributes like rows, cols, and placeholder.

The <select> Element:

Creates a dropdown list from which users can select one or more options. It should contain one or more <option> tags that define the choices.

The <option> Element:

Defines an individual option in a dropdown list created by the <select> tag. It can have attributes like value and selected.

Hypertext Markup Language (HTML)

Forms in HTML

The <fieldset> Element:

Groups related elements within a form, providing a visual and semantic way to organize the form. Often used with a <legend> for labeling.

The <legend> Element:

Provides a caption for the content of a <fieldset>. It helps to describe the purpose of the grouped form controls.

The <button> Element:

Defines a clickable button. It can be used as a submit button or for any custom actions. It allows for more complex content (like images or text) than the <input type="submit"> button.

Hypertext Markup Language (HTML)

Forms in HTML

The <datalist> Element:

Provides a set of predefined options for an <input> element, which can be displayed as suggestions while typing.

- Provides a list of suggested options that the user can choose from while typing into an associated <input> field.
- The <datalist> element is linked to an <input> element via the list attribute of the <input> element.
- Used when you want to offer suggestions, but still allow the user to input something that is not in the predefined list.

The <output> Element:

Represents the result of a calculation or user action, such as the result of a form submission.

```
<!DOCTYPE html>
<html>

<head><title>Forms in HTML</title></head>

<body>

<form action="/submitform.jsp" method="POST">
<label for="city">City: </label>
<input type="text" id="city" name="city"><br>

<label for="pwd">Password:</label>
<input type="password" id="pwd" name="pwd"><br>

<label for="male">Male</label>
<input type="radio" id="male" name="gender" value="M">

<label for="female">Female</label>
<input type="radio" id="female" name="gender" value="F">
<br>

<label for="email">Email Id:</label>
<input type="email" id="email"><br>

<input type="number" max="30" min="10" step="2">
<label for = "WT">Web Technology</label>
<input type="checkbox" name="course" id="WT"><br>

<input type="date"><br>
<input type="submit" value="Submit">
<input type="reset" value="Reset">


<textarea rows="5" cols="50"></textarea>
<select>
<option>WebTechnology</option>
<option>Database Management System</option>
</select>

<input list="courses">
<datalist id="courses">
<option>WebTechnology</option>
<option>Database Management System</option>
</datalist>
</form>

</body>
</html>
```

```
<!DOCTYPE html>
<head>
  <title>Input Types Example</title>
</head>
<body>
  <h1>HTML Input Fields Demonstration</h1>
  <form action="/submit" method="POST" enctype="multipart/form-data">

    <!-- Text Input -->
    <label for="textInput">Text Input:</label>
    <input type="text" id="textInput" name="textInput" placeholder="Enter text" required><br><br>

    <!-- Password Input -->
    <label for="passwordInput">Password Input:</label>
    <input type="password" id="passwordInput" name="passwordInput" placeholder="Enter
password" required><br><br>

    <!-- Submit Button -->
    <input type="submit" value="Submit">

    <!-- Reset Button -->
    <input type="reset" value="Reset"><br><br>

    <!-- Radio Buttons -->
    <label>Choose an option:</label><br>
    <input type="radio" id="option1" name="options" value="Option 1" required>
    <label for="option1">Option 1</label><br>
    <input type="radio" id="option2" name="options" value="Option 2">
    <label for="option2">Option 2</label><br><br>

    <!-- Checkbox -->
    <label for="checkbox">Select your interests:</label><br>
    <input type="checkbox" id="interest1" name="interests" value="Interest 1">
    <label for="interest1">Interest 1</label><br>
    <input type="checkbox" id="interest2" name="interests" value="Interest 2">
    <label for="interest2">Interest 2</label><br><br>

    <!-- Email Input -->
    <label for="emailInput">Email Input:</label>
    <input type="email" id="emailInput" name="emailInput" placeholder="Enter your email"
required><br><br>

    <!-- Number Input -->
    <label for="numberInput">Number Input:</label>
    <input type="number" id="numberInput" name="numberInput" min="1" max="100" step="1"
required><br><br>
```

```
<!-- Date Input -->
<label for="dateInput">Date Input:</label>
<input type="date" id="dateInput" name="dateInput" required><br><br>

<!-- Time Input -->
<label for="timeInput">Time Input:</label>
<input type="time" id="timeInput" name="timeInput" required><br><br>

<!-- File Upload -->
<label for="fileInput">File Input:</label>
<input type="file" id="fileInput" name="fileInput" accept=".jpg,.png,.pdf" required><br><br>

<!-- Search Input -->
<label for="searchInput">Search Input:</label>
<input type="search" id="searchInput" name="searchInput" placeholder="Search..."><br><br>

<!-- Telephone Input -->
<label for="telInput">Telephone Input:</label>
<input type="tel" id="telInput" name="telInput" placeholder="Enter your phone number"><br><br>

<!-- URL Input -->
<label for="urlInput">URL Input:</label>
<input type="url" id="urlInput" name="urlInput" placeholder="Enter a URL" required><br><br>

<!-- Color Input -->
<label for="colorInput">Color Input:</label>
<input type="color" id="colorInput" name="colorInput" value="#ff0000"><br><br>

<!-- Tel Input -->
<label for="telInput">Telephone Input:</label>
<input type="tel" id="telInput" name="telInput" placeholder="Enter your phone number"><br><br>

<!-- Fieldset and Legend -->
<fieldset>
  <legend>Personal Information</legend>

  <!-- Label and Textarea -->
  <label for="bio">Biography:</label><br>
  <textarea id="bio" name="bio" rows="4" cols="50" placeholder="Tell us about
yourself..."></textarea><br><br>

  <!-- Select and Option -->
  <label for="gender">Gender:</label>
  <select id="gender" name="gender">
    <option value="male">Male</option>
    <option value="female">Female</option>
    <option value="other">Other</option>
  </select><br><br>
```

```
    </fieldset>

    <!-- Datalist -->
    <label for="city">City:</label>
    <input list="cities" id="city" name="city" placeholder="Choose or type your city">
    <datalist id="cities">
      <option value="New York">
      <option value="Los Angeles">
      <option value="Chicago">
      <option value="Houston">
      <option value="Miami">
    </datalist><br><br>

    <!-- Button -->
    <button type="submit">Submit</button>

    <!-- Output -->
    <p>Form submission status: <output name="result" for="form"></output></p>

  </form>
</body>
</html>
```

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-8

Today's Target

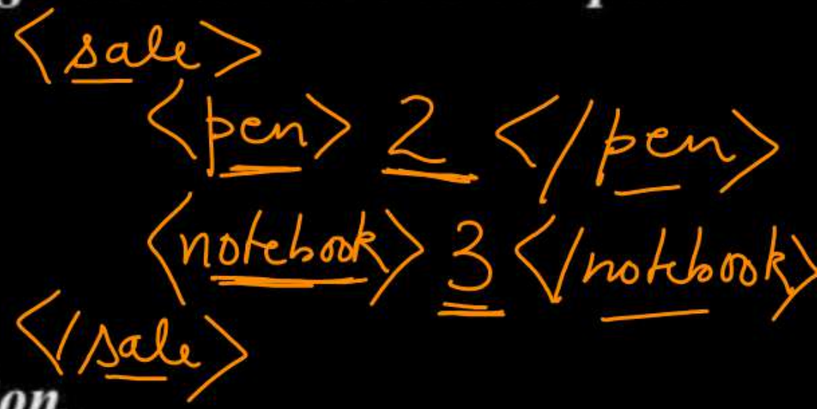
- eXtensible Markup Language (XML)
  - What is XML?
  - HTML and XML
  - XML Document Structure
  - Well Formed XML
  - AKTU PYQs

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

eXtensible Markup Language (XML)

"eXtensible Markup Language (XML) is a markup language that allows users to store and share data in a structured format. XML is used for information exchange between diverse computer systems and applications."



- *XML is used to describe structured data or information.*
- *Tags are added to the document to provide the extra information.*
- *The data are intended to be used by machines or people.*
- *An XML document separates the contents from their presentation.*
- *XML documents are used to share data among applications over the Web.*

eXtensible Markup Language (XML)

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<student>
    <name>John Doe</name>
    <age>21</age>
    <branch>Computer Science</branch>
    <contact>
        <email>john.doe@example.com</email>
        <phone>123-XXX-7890</phone>
    </contact>
</student>
```

eXtensible Markup Language (XML)

Difference Between HTML and XML

- HTML documents describe how data should appear on the browser's screen they carry no information about the data where as XML documents describe the meaning of data.
- HTML is used to design web pages specifying the content and its appearance/ presentation where as XML is used as a primary means to store and transfer structured data over the web.
- HTML has predefined fixed set of tags that tell a browser how to display the document where as XML allows us to define new tags and use them to give a reader some idea what some of the data means.
- HTML tags have a fixed meaning and browsers know what it is where as XML tags are different for different applications, and users know what they mean.

eXtensible Markup Language (XML)

XML Document Structure

An XML document consists of the following parts:

- Prolog
- Body

Prolog

The prolog may contain the following components:

XML Declaration:

Indicates the version of XML and the character encoding used.

Example: <?xml version="1.0" encoding="UTF-8" standalone="yes"?>

Attributes:

version: Specifies the XML version (usually "1.0").

encoding: Defines the character encoding (commonly "UTF-8").

standalone: Indicates whether the document relies on external markup declarations (values: "yes" or "no").

eXtensible Markup Language (XML)

XML Document Structure

Prolog

The prolog may contain the following components:

Processing Instructions:

Used to pass specific instructions or parameters to applications that process the XML document.

Example: <?xml-stylesheet type="text/xsl" href="style.xsl"?>

Syntax: Begins with <? and ends with ?>.

Comments:

Used to add notes or explanations within the XML document, which are ignored by the parser.

Example: <!-- This is a comment -->

Syntax: Begins with <!-- and ends with -->.

eXtensible Markup Language (XML)

XML Document Structure

Prolog

The prolog may contain the following components:

- **Document Type Declaration (DTD):**
  - Specifies the logical structure of the XML document, including the tags that can be used and their relationships.
  - Example: <!DOCTYPE note SYSTEM "Note.dtd">
  - Purpose: Helps with validation, ensuring that the XML adheres to a defined structure.

eXtensible Markup Language (XML)

XML Document Structure

Body

The body contains the actual content of the XML document, marked up by tags.

Key Characteristics:

Single Root Element: The document must have one root element that encapsulates all other elements.

Example: In the XML <note><to>Tove</to></note>, <note> is the root element.

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <content>Attend the classes of Web Technology regularly!</content>
</note>
```

Nesting of Elements: The root element can contain other elements, which may in turn contain further elements, creating a hierarchical structure.

eXtensible Markup Language (XML)

Well Formed XML

An XML document is said to be well-formed if it follows all the basic rules. An XML parser is used to check that all the rules have been obeyed.

<sale customer="Anuj">

</sale>

Basic Rules:

- XML tags are case sensitive.
- All start tags must have end tags.
- Elements must be properly nested.
- XML declaration is the first statement.
- Every document must contain one and exactly one root element.
- Attribute values must have quotation marks.
- Certain characters are reserved for parsing.

  For Example: < , >

```
<users>
    <name>Abc</name>
    <username>@abc</username>
    <followers>200</followers>
    <posts>34</posts>
</users>
```

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-9

Today's Target

- eXtensible Markup Language (XML)
- Validating XML
- Dcoument Type Definition (DTD)
- XML Schema Definition (XSD)
- AKTU PYQs

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

eXtensible Markup Language (XML)

Valid XML

- *are well formed*

  obey basic well-formedness rules.

- *comply with rules specified in DTD or schema.*

  rules usually specify the name and content of the element that can occur in the valid documents.

eXtensible Markup Language (XML)

Document Type Definition (DTD)

"The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements"

Syntax:

<!DOCTYPE element DTD identifier [declaration1 declaration2 .....]>

In the above syntax,

- The DTD starts with <!DOCTYPE delimiter.
- The element specified after <!DOCTYPE indicates the root element that will contain the XML document's content.
- DTD identifier is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called *External Subset*.
- The square brackets [ ] enclose an optional list of declarations (elements, attributes, entities, etc.) called the *Internal Subset*.

eXtensible Markup Language (XML)

Document Type Definition (DTD)

-Internal DTD

An internal DTD is defined directly within the XML document itself. It specifies the structure and rules for that particular XML file.

-External DTD

An external DTD is stored in a separate file and referenced by the XML document. It allows multiple XML files to share the same structure and validation rules.

eXtensible Markup Language (XML)

Document Type Definition (DTD)

-Components of XML documents

From a DTD perspective, all XML documents are composed of several key components:

- Elements
- Attributes
- Entities
- Parsed Character Data (PCDATA)
- Character Data (CDATA)

eXtensible Markup Language (XML)

Document Type Definition (DTD)

-Components of XML documents

- Elements:

  Elements are the fundamental building blocks of an XML document. They define the structure of the document and encapsulate data. Each element is marked by a start tag and an end tag with content between them.

  Example:

  ```
  <name>John Doe</name>
  ```

- Attributes:

  Attributes provide additional information about elements. They appear inside the opening tag of an element and have a name-value pair format. Attributes help describe properties of elements.

  Example:

  ```
  <person gender="male">
    <name>John Doe</name>
  </person>
  ```

eXtensible Markup Language (XML)

Document Type Definition (DTD)

-Components of XML documents

Entities:

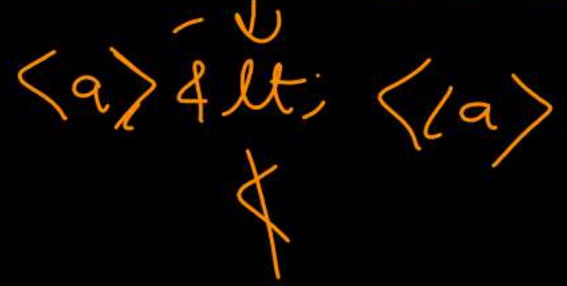
Entities are used to define shortcuts to larger blocks of text or special characters. There are predefined entities (like < for < and & for &), as well as custom entities that can be declared in the DTD for reuse.

Example:

```
<!ENTITY author "John Doe">
```

Usage in XML:

```
<book>&author;</book>
```



PCDATA (Parsed Character Data):

PCDATA represents the data that is parsed by the XML processor, meaning that any special characters (like & or <) are recognized and processed. This is the normal text between XML tags.

Example:

```
<title>The &amp;amp; Story</title>
```

eXtensible Markup Language (XML)

Document Type Definition (DTD)

-Components of XML documents

CDATA (Character Data):

CDATA is a block of text that is not parsed by the XML processor. Everything within a CDATA section is treated as literal text, meaning special characters or reserved XML symbols are not interpreted as XML code.

Example:

```
<message>
  <![CDATA[This is <important> information]]>
</message>
```

eXtensible Markup Language (XML)

Document Type Definition (DTD)

-DTD - Elements

Elements with Parsed Character Data

Elements with only parsed character data are declared with #PCDATA inside parentheses:

<!ELEMENT element-name (#PCDATA)>

Example:

<!ELEMENT from (#PCDATA)>

Elements with Children (sequences)

Elements with one or more children are declared with the name of the children elements inside parentheses:

<!ELEMENT element-name (child1,child2,...)>

Example:

<!ELEMENT note (to, from, heading, body)>

eXtensible Markup Language (XML)

Document Type Definition (DTD)

-DTD - Elements

Elements with Children (sequences)

Declaring Only One Occurrence of an Element: <!ELEMENT note (message)>
The example above declares that the child element "message" must occur once, and only once inside the "note" element.

Declaring Minimum One Occurrence of an Element: <!ELEMENT note (message+)>
The + sign in the example above declares that the child element "message" must occur one or more times inside the "note" element.

Declaring Zero or More Occurrences of an Element: <!ELEMENT note (message*)>
The * sign in the example above declares that the child element "message" can occur zero or more times inside the "note" element.

Declaring Zero or One Occurrences of an Element : <!ELEMENT note (message?)>
The ? sign in the example above declares that the child element "message" can occur zero or one time inside the "note" element.

eXtensible Markup Language (XML)

Document Type Definition (DTD)

-DTD – Attributes

In a DTD, attributes are declared with an ATTLIST declaration.

Declaring Attributes

An attribute declaration has the following syntax:

```
<!ATTLIST element-name attribute-name attribute-type attribute-value>
```

DTD example:

```
<!ATTLIST payment type CDATA "check">
```

XML example:

```
<payment type="check" />
```

eXtensible Markup Language (XML)

Document Type Definition (DTD)

-DTD – Attributes

In a DTD, attributes are declared with an ATTLIST declaration.

Declaring Attributes

An attribute declaration has the following syntax:

<!ATTLIST element-name attribute-name attribute-type attribute-value>

The attribute-value can be one of the following:

value	*The default value of the attribute*
#REQUIRED	*The attribute is required*
#IMPLIED	*The attribute is optional*
#FIXED value	*The attribute value is fixed*

eXtensible Markup Language (XML)

Document Type Definition (DTD)

Internal DTD

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>

<!DOCTYPE address
[ <!ELEMENT address (name, company, phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)> ]>

<address>
<name>Samar Singh</name>
<company>AOL</company>
<phone>99XXXX6622</phone>
</address>
```

Valid

External DTD

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
<name>Samar Singh</name>
<company>AOL</company>
<phone>99XXXX6622</phone>
</address>
```

The content of the DTD file **address.dtd** is as follows –

```
<!ELEMENT address (name, company, phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- EXTERNAL DTD -->
<!--<!DOCTYPE email SYSTEM "email.dtd">-->

<!-- INTERNAL DTD -->
<!DOCTYPE email
[
<!ELEMENT email (header,body, attachments?)>
<!ELEMENT header (from,to,subject,date)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT date (#PCDATA)>

<!ELEMENT body (#PCDATA)>

<!ELEMENT attachments (attachment*)>
<!ELEMENT attachment (filename,filetype)>
<!ELEMENT filename (#PCDATA)>
<!ELEMENT filetype (#PCDATA)>

<!ATTLIST email id CDATA #REQUIRED>
<!ATTLIST from email CDATA #REQUIRED>
<!ATTLIST to email CDATA #REQUIRED>
<!ATTLIST attachment size CDATA #IMPLIED>
]
>
<email id="email001">
  <header>
    <from email="john.doe@example.com">John Doe</from>
    <to email="jane.doe@example.com">Jane Doe</to>
    <subject>Meeting Reminder</subject>
    <date>2024-09-25</date>
  </header>
  <body>
    This is a reminder for our meeting scheduled tomorrow at 10 AM.
  </body>
  <attachments>
    <attachment size="2MB">
      <filename>meeting_agenda.pdf</filename>
      <filetype>application/pdf</filetype>
    </attachment>
  </attachments>
</email>
```

```
<!ELEMENT email (header, body, attachments?)>
<!ELEMENT header (from, to, subject, date)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT date (#PCDATA)>

<!ELEMENT body (#PCDATA)>

<!ELEMENT attachments (attachment*)>
<!ELEMENT attachment (filename, filetype)>
<!ELEMENT filename (#PCDATA)>
<!ELEMENT filetype (#PCDATA)>

<!ATTLIST email id CDATA #REQUIRED>
<!ATTLIST from email CDATA #REQUIRED>
<!ATTLIST to email CDATA #REQUIRED>
<!ATTLIST attachment size CDATA #IMPLIED>
```

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-10

Today's Target

- eXtensible Markup Language (XML)
- Validating XML
  - XML Schema,
  - XML Schema Defintion (XSD)
  - DTD and XSD
  - Defining Elements and Attribues in XSD
- AKTU PYQs

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

eXtensible Markup Language (XML)

XML Schema

"An XML Schema defines the structure, content, and data types of an XML document. It serves as a blueprint that specifies which elements and attributes can appear, their relationships, and the type of data they hold."

XML Schema Definition (XSD)

"XML Schema is the concept or the standard, while XSD is a specific language used to create those schemas. So, XML Schema is defined using XSD."

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

DTD and XSD

- XSD schemas are themselves XML documents, making them easier to work with in XML-based environments, whereas DTDs use a separate, non-XML syntax.
- XSDs were standardized after DTDs and offer more detailed information about the document, allowing for richer document structure definitions.
- Unlike DTDs, XSDs provide a wide variety of built-in data types such as string, decimal, integer, boolean, date, and time, enabling more precise data validation.
- Additionally, XSDs divide elements into simple and complex types, offering greater flexibility in defining document structure compared to DTDs.

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

The <schema> Element

The <schema> element is the root element of every XML Schema:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
...
</xs:schema>
```

xmlns:xs="http://www.w3.org/2001/XMLSchema" indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace and should be prefixed with 'xs:'.

[NOTE: In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications. So, *XML Namespaces* provide a method to avoid element name conflicts.
Syntax: xmlns: prefix = "URI".]

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

XSD Simple Elements

A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.

Simple XML Elements

```
<lastname>Sharma</lastname>
<age>16</age>
```

Simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
```

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

XSD Simple Elements

XML Schema has a lot of built-in data types. The most common types are:

xs:string, xs:decimal, xs:integer, xs:Boolean, xs:date, and xs:time

Default and Fixed Values for Simple Elements

A default value is automatically assigned to the element when no other value is specified.

```
<xs:element name="color" type="xs:string" default="red"/>
```

A fixed value is also automatically assigned to the element, and it can not be changed.

```
<xs:element name="color" type="xs:string" fixed="red"/>
```

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

XSD Complex Elements

A complex element is an XML element that contains other elements and/or attributes.

Complex XML Element

```
<employee id="E101">
 <firstname>John</firstname>
 <lastname>Smith</lastname>
</employee>
```

Complex element definition:

```
<xs:element name="employee">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="firstname" type="xs:string"/>
   <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
 <xs:attribute name="id" type="xs:string"/>
</xs:element>
```


[NOTE: The child elements, "firstname" and "lastname", are surrounded by the <sequence> indicator. This means that the child elements must appear in the same order as they are declared.]

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

XSD Complex Elements

A complex element is an XML element that contains other elements and/or attributes.

Complex XML Element

```
<from
email="john.doe@example.com">
John Doe
</from>
```

Complex element definition:

```
<xs:element name="from">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="email" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```


[NOTE: <simpleContent> is used for complex types that have no child elements but can contain text and attributes. It allows for extension of simple types by adding attributes.]

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

XSD Elements

Occurrence Indicators

Occurrence indicators are used to define how often an element can occur.

[NOTE: The default value for maxOccurs and minOccurs is 1.]

maxOccurs Indicator

The 'maxOccurs' indicator specifies the maximum number of times an element can occur:

```
<xs:element name="courses" type="xs:string" maxOccurs="6"/>
```

minOccurs Indicator

The 'minOccurs' indicator specifies the minimum number of times an element can occur:

```
<xs:element name="courses" type="xs:string" maxOccurs="6" minOccurs="5"/>
```

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

XSD Attributes

An attribute of an XML element is a name-value pair that provides additional information about that element. They are specified within the opening tag of an element.

XML element with an attribute:

```
<lastname lang="EN">Smith</lastname>
```

Attribute definition:

```
<xs:attribute name="lang" type="xs:string"/>
```

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

XSD Attributes

Default and Fixed Values for Attributes

A default value is automatically assigned to the attribute when no other value is specified.

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

A fixed value is also automatically assigned to the attribute, and it can not be changed.

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

Optional and Required Attributes

Attributes are optional by default. To specify that the attribute is required, use the "use" attribute:

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```

eXtensible Markup Language (XML)

XML Schema Definition (XSD)

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="address">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="name" type="xs:string"/>
                <xs:element name="email" type="xs:string"/>
                <xs:element name="phone" type="xs:string"/>
                <xs:element name="birthday" type="xs:date"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="email">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="header">
          <xs:complexType>
                <xs:sequence>
                      <xs:element name="from">
                            <xs:complexType>
                            <xs:simpleContent>
                                  <xs:extension base="xs:string">
                                        <xs:attribute name="email" type="xs:string"/>
                                  </xs:extension>
                            </xs:simpleContent>
                            </xs:complexType>
                      </xs:element>
                      <xs:element name="to">
                            <xs:complexType>
                            <xs:simpleContent>
                                  <xs:extension base="xs:string">
                                        <xs:attribute name="email" type="xs:string"/>
                                  </xs:extension>
                            </xs:simpleContent>
                            </xs:complexType>
                      </xs:element>
                      <xs:element name="subject" type="xs:string"/>
                      <xs:element name="date" type="xs:date"/>
                </xs:sequence>
          </xs:complexType>
    </xs:element>
    <xs:element name="body" type="xs:string"/>
    <xs:element name="attachments">
          <xs:complexType>
                <xs:sequence>
                      <xs:element name="attachment">
                            <xs:complexType>
                                  <xs:sequence>
                                        <xs:element name="filename" type="xs:string"/>
                                        <xs:element name="filetype" type="xs:string"/>
                                  </xs:sequence>
                                  <xs:attribute name="size" type="xs:string"/>
                            </xs:complexType>
                      </xs:element>
                </xs:sequence>
          </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="id" type="xs:string"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>

<email
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="myemail.xsd"
id="email001">
  <header>
    <from email="john.doe@example.com">John Doe</from>
    <to email="jane.doe@example.com">Jane Doe</to>
    <subject>Meeting Reminder</subject>
    <date>2024-09-25</date>
  </header>
  <body>
    This is a reminder for our meeting scheduled tomorrow at 10 AM.
  </body>
  <attachments>
    <attachment size="2MB">
      <filename>meeting_agenda.pdf</filename>
      <filetype>application/pdf</filetype>
    </attachment>
  </attachments>
</email>
```

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-11

Today's Target

- eXtensible Markup Language (XML)
  - Presenting and using XML
    - XSLT
      - Elements
  - Object Model
  - AKTU PYQs

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

eXtensible Markup Language (XML)

Presenting and using XML

eXtensible Stylesheet Language Transformation (XSLT)

"XSLT is a language used to transform one xml document into another, often an html document. A program is used that takes as input one xml document and produces as output another."

- If the resulting document is in html, it can be viewed by a web browser.
- This is a good way to display xml data.

address.xml

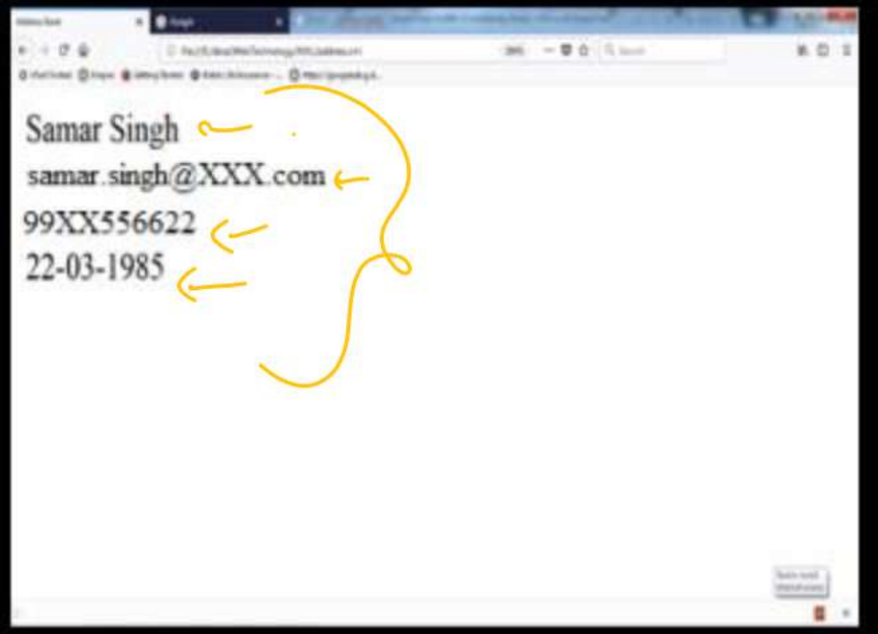
```
<?xml version="1.0" encoding = "UTF-8"?>
<?xml-stylesheet type="text/xml" href="transform.xsl"?>
<address>
    <name>Samar Singh</name>
    <email>samar.singh@XXX.com</email>
    <phone>99XX556622</phone>
    <birthday>22-03-1985</birthday>
</address>
```

eXtensible Markup Language (XML)

eXtensible Stylesheet Language Transformation (XSLT)

transform.xsl

```
<?xml version="1.0" encoding = "UTF-8"?>
    <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="address">
            <html><head><title>Address Book</title></head>
            <body>
                <xsl:value-of select="name"/>
                <br/><xsl:value-of select="email"/>
                <br/><xsl:value-of select="phone"/>
                <br/><xsl:value-of select="birthday"/>
            </body>
            </html>
        </xsl:template>
    </xsl:stylesheet>
```



eXtensible Markup Language (XML)

eXtensible Stylesheet Language Transformation (XSLT)

Elements for XML Transformation

<xsl:template>: Defines a template that matches a specific part of an XML document and transforms it according to the rules inside the template.

Example: <xsl:template match="/"> <html><body><h2>Book List</h2></body></html> </xsl:template>

<xsl:for-each>: Definition: Iterates over a set of nodes, applying the transformation inside the loop for each node.

Example: <xsl:for-each select="books/book"> <xsl:value-of select="title" /> </xsl:for-each>

eXtensible Markup Language (XML)

eXtensible Stylesheet Language Transformation (XSLT)

Elements for XML Transformation

<xsl:if>: Definition: Conditionally applies a transformation if a specified condition is true.

Example: <xsl:if test="price > 30"> This book is expensive! </xsl:if>

xsl:value-of>: Definition: Extracts and outputs the value of a specific XML element or attribute.

Example: <xsl:value-of select="author" />

eXtensible Markup Language (XML)

eXtensible Stylesheet Language Transformation (XSLT)

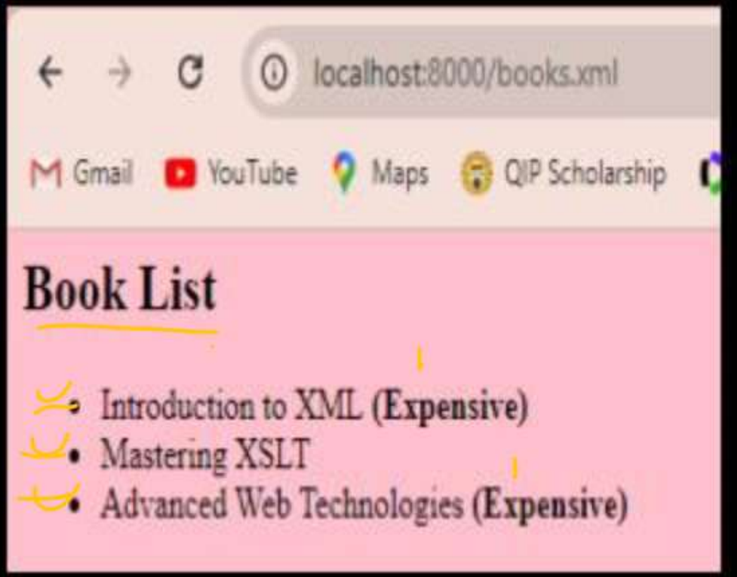
Elements for XML Transformation - Example

books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="books.xsl"?>
<books>
  <book>
    <title>Introduction to XML</title>
    <author>John Doe</author>
    <price>45</price>
  </book>
  <book>
    <title>Mastering XSLT</title>
    <author>Jane Smith</author>
    <price>25</price>
  </book>
  <book>
    <title>Advanced Web Technologies</title>
    <author>Tom Brown</author>
    <price>50</price>
  </book>
</books>
```

books.xsl

```
<?xml version="1.0" encoding = "UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
  <body bgcolor="pink">
   <h2>Book List</h2>
   <ul>
    <xsl:for-each select="books/book">
     <li>
      <xsl:value-of select="title" />
      <xsl:if test="price &gt; 30">
       <strong> (Expensive)</strong>
      </xsl:if>
     </li>
    </xsl:for-each>
   </ul>
  </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```



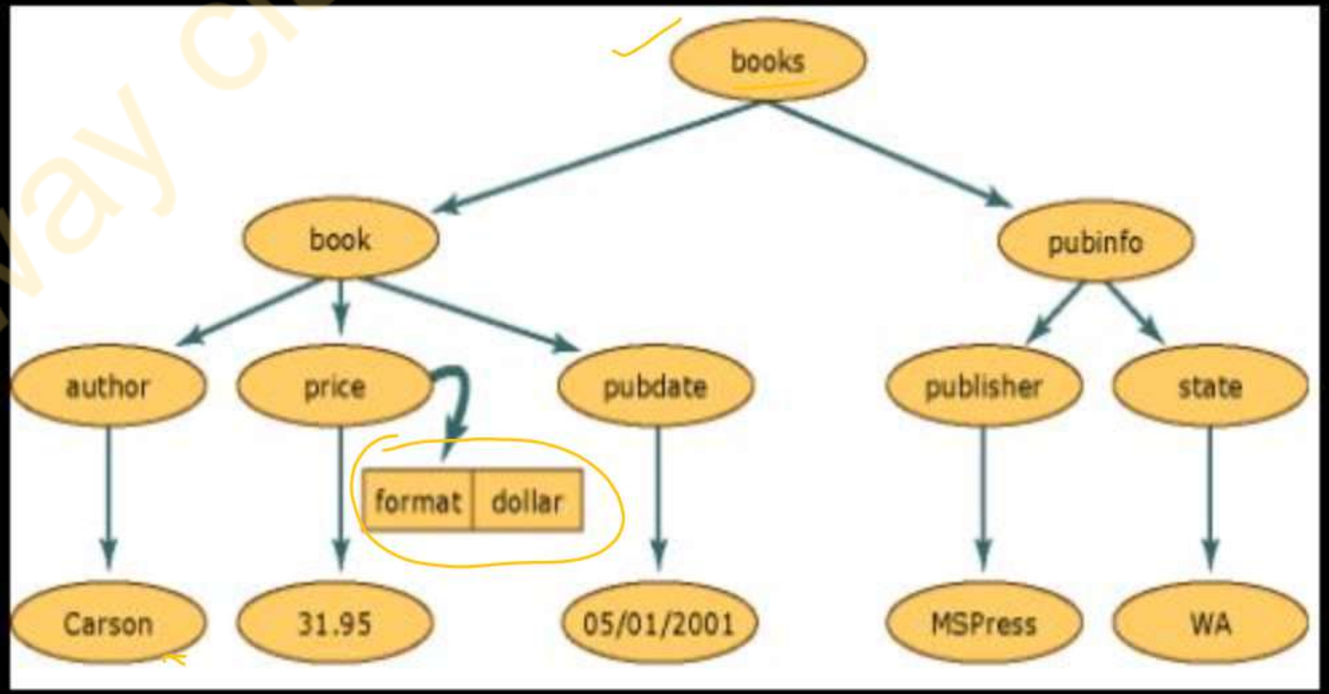
eXtensible Markup Language (XML)

Object Model

"The XML object model represents an XML document as a tree-like structure, where each element is a node. The relationships between these nodes reflect the nesting of elements, and each XML element, attribute, and text node is treated as an individual object."

Example:

```
<?xml version="1.0"?>
 <books>
  <book>
    <author>Carson</author>
    <price format="dollar">31.95</price>
    <pubdate>05/01/2001</pubdate>
  </book>
  <pubinfo>
    <publisher>MSPress</publisher>
    <state>WA</state>
  </pubinfo>
 </books>
```



eXtensible Markup Language (XML)

Object Model

- **Node Types:** In the object model, there are several node types, including element nodes, attribute nodes, text nodes, and comment nodes, each representing different components of the XML.

- **Properties and Methods:** Each node object in the object model has properties (such as name, value, and type) and methods (like getChildNodes(), appendChild(), etc.) that allow for manipulation and retrieval of data.

- **Access and Manipulation:** The object model provides a way to access and manipulate XML data programmatically, enabling dynamic modifications and querying of the document.

books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="books.xsl"?>
<books>
 <book>
  <title>Introduction to XML</title>
  <author>John Doe</author>
  <price>45</price>
 </book>
 <book>
  <title>Mastering XSLT</title>
  <author>Jane Smith</author>
  <price>25</price>
 </book>
 <book>
  <title>Advanced Web Technologies</title>
  <author>Tom Brown</author>
  <price>50</price>
 </book>
</books>
```

books.xsl

```
<?xml version="1.0" encoding = "UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
  <body bgcolor="pink">
   <h2>Book List</h2>
   <ul>
    <xsl:for-each select="books/book">
     <li>
      <xsl:value-of select="title" />
      <xsl:if test="price &gt; 30">
       <strong> (Expensive)</strong>
      </xsl:if>
     </li>
    </xsl:for-each>
   </ul>
  </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Student_marks.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="viewmarksheet.xsl"?>
<students>
  <student>
    <rollNumber>101</rollNumber>
    <marks>
      <subject name="Database Management System">85</subject>
      <subject name="Web Technology">78</subject>
      <subject name="Design and Analysis of Algorithm">82</subject>
      <subject name="Object Oriented System Design with C++">90</subject>
      <subject name="Machine Learning Techniques">75</subject>
    </marks>
    <totalMarks>410</totalMarks>
    <percentage>82.00</percentage>
  </student>
  <student>
    <rollNumber>102</rollNumber>
    <marks>
      <subject name="Database Management System">88</subject>
      <subject name="Web Technology">74</subject>
      <subject name="Design and Analysis of Algorithm">79</subject>
      <subject name="Object Oriented System Design with C++">92</subject>
      <subject name="Machine Learning Techniques">81</subject>
    </marks>
    <totalMarks>414</totalMarks>
    <percentage>82.80</percentage>
  </student>
  <student>
    <rollNumber>103</rollNumber>
    <marks>
      <subject name="Database Management System">76</subject>
      <subject name="Web Technology">82</subject>
      <subject name="Design and Analysis of Algorithm">84</subject>
      <subject name="Object Oriented System Design with C++">88</subject>
      <subject name="Machine Learning Techniques">77</subject>
    </marks>
    <totalMarks>407</totalMarks>
    <percentage>81.40</percentage>
  </student>
  <student>
    <rollNumber>104</rollNumber>
    <marks>
      <subject name="Database Management System">90</subject>
      <subject name="Web Technology">85</subject>
      <subject name="Design and Analysis of Algorithm">80</subject>
      <subject name="Object Oriented System Design with C++">95</subject>
      <subject name="Machine Learning Techniques">89</subject>
    </marks>
    <totalMarks>439</totalMarks>
```

```
    <percentage>87.80</percentage>
  </student>
  <student>
    <rollNumber>105</rollNumber>
    <marks>
      <subject name="Database Management System">82</subject>
      <subject name="Web Technology">78</subject>
      <subject name="Design and Analysis of Algorithm">76</subject>
      <subject name="Object Oriented System Design with C++">89</subject>
      <subject name="Machine Learning Techniques">80</subject>
    </marks>
    <totalMarks>405</totalMarks>
    <percentage>81.00</percentage>
  </student>
  <student>
    <rollNumber>106</rollNumber>
    <marks>
      <subject name="Database Management System">87</subject>
      <subject name="Web Technology">91</subject>
      <subject name="Design and Analysis of Algorithm">80</subject>
      <subject name="Object Oriented System Design with C++">94</subject>
      <subject name="Machine Learning Techniques">76</subject>
    </marks>
    <totalMarks>428</totalMarks>
    <percentage>85.60</percentage>
  </student>
  <student>
    <rollNumber>107</rollNumber>
    <marks>
      <subject name="Database Management System">75</subject>
      <subject name="Web Technology">88</subject>
      <subject name="Design and Analysis of Algorithm">83</subject>
      <subject name="Object Oriented System Design with C++">90</subject>
      <subject name="Machine Learning Techniques">72</subject>
    </marks>
    <totalMarks>408</totalMarks>
    <percentage>81.60</percentage>
  </student>
  <student>
    <rollNumber>108</rollNumber>
    <marks>
      <subject name="Database Management System">80</subject>
      <subject name="Web Technology">84</subject>
      <subject name="Design and Analysis of Algorithm">86</subject>
      <subject name="Object Oriented System Design with C++">87</subject>
      <subject name="Machine Learning Techniques">78</subject>
    </marks>
    <totalMarks>415</totalMarks>
    <percentage>83.00</percentage>
  </student>
  <student>
```

```
    <percentage>86.20</percentage>
  </student>
  <student>
    <rollNumber>110</rollNumber>
    <marks>
      <subject name="Database Management System">86</subject>
      <subject name="Web Technology">82</subject>
      <subject name="Design and Analysis of Algorithm">85</subject>
      <subject name="Object Oriented System Design with C++">90</subject>
      <subject name="Machine Learning Techniques">84</subject>
    </marks>
    <totalMarks>427</totalMarks>
    <percentage>85.40</percentage>
  </student>
</students>
```

viewmarksheet.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<head>
  <title>Student Marksheet</title>
</head>
<body>
<xsl:for-each select="students/student">
  <h1>Marksheet of Student</h1>
  <h2>Roll Number: <xsl:value-of select="rollNumber"/></h2>
  <table border="1">
    <thead>
      <tr>
        <th>Subject</th>
        <th>Marks Obtained</th>
      </tr>
    </thead>
    <tbody>
<xsl:for-each select="marks/subject">
      <tr>
        <td><xsl:value-of select="@name"/></td>
        <td><xsl:value-of select="."/></td>
```

```
        </tr>
</xsl:for-each>
        <tr>
          <td><b>Total Marks</b></td>
          <td><xsl:value-of select="totalMarks"/></td>
        </tr>
        <tr>
          <td><b>Percentage</b></td>
          <td><xsl:value-of select="percentage"/></td>
        </tr>
      </tbody>
    </table>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

AKTU

B.Tech 5th Sem

GATEWAY CLASSES

CS IT & CS Allied

Web Technology

UNIT-1 Lecture-12

Today's Target

- eXtensible Markup Language (XML)
  - XML Processors
    - XML Parsers
      - DOM Parser
      - SAX Parser
  - AKTU PYQs

By Amol Sharma sir

- Pursuing Ph.D. from IIT (BHU)
- Wipro Certified Faculty (WCF)

eXtensible Markup Language (XML)

XML Processors

- When a software program reads an XML document and takes actions accordingly, this is called processing the XML.
- Any program that can read and process XML documents is known as an XML processor.
- The most fundamental XML processor reads an XML document and converts it into an internal representation for other programs to use. This is called an XML parser, and it is an important component of every XML processing program.
- XML processors ensure the XML document is well-formed and, in the case of a validating processor, checks its validity against a defined schema or DTD.
- There are two types of XML parsers:
  - Document Object Model (DOM) Parser
  - Simple API for XML (SAX) Parser

eXtensible Markup Language (XML)

XML Processors

Example:

XML File:

```
<?xml version="1.0" encoding = "UTF-8"?>
<address>
<name>Samar Singh</name>
<email>samar.singh@XXX.com</email>
<phone countryCode="+91">99XX556622</phone>
<gender>Male</gender>
<birthday>22-03-1985</birthday>
</address>
```

Program Output:

Name: Mr. Samar Singh

Email: samar.singh@XXX.com

Phone No: +91 99XX556622

Gender: Male

Birthday: 22-03-1985

eXtensible Markup Language (XML)

XML Parsers

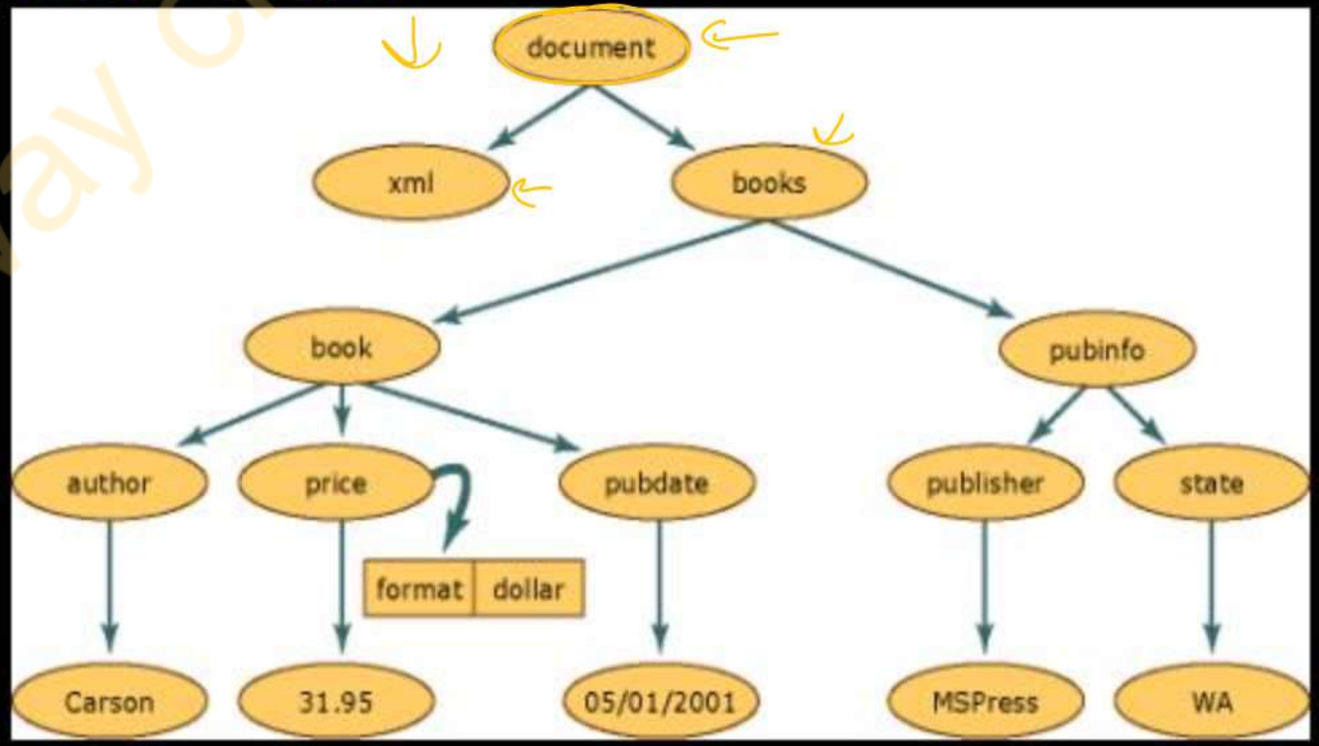
XML DOM Parser

"XML DOM Parser is a tool that reads XML documents and loads them into memory, creating a tree structure (Document Object Model or DOM structure) that represents the entire document."

XML File:

```
<?xml version="1.0"?>
 <books>
  <book>
    <author>Carson</author>
    <price format="dollar">31.95</price>
    <pubdate>05/01/2001</pubdate>
  </book>
  <pubinfo>
    <publisher>MSPress</publisher>
    <state>WA</state>
  </pubinfo>
 </books>
```

DOM Structure:



eXtensible Markup Language (XML)

XML Parsers

XML DOM Parser

- In-memory Representation: The DOM parser loads the entire XML document into memory and represents it as a tree of nodes (elements, attributes, text, etc.). Each element, attribute, and text in the XML document becomes an object (node) in the DOM tree.
- Full Access to Document: Once the document is loaded, programs can access and modify the structure and content of the document through standard DOM methods.
- Suitable for Small to Medium XML Files: Since it loads the entire document into memory, it's ideal for small to medium-sized XML documents, but not for very large files.

eXtensible Markup Language (XML)

XML Parsers

XML DOM Parser in Java

```
import java.io.File;
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMDemo {
    public static void main(String[] args) throws Exception {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = factory.newDocumentBuilder();
        File xmlFile = new File("address.xml");
        Document xmlDoc = docBuilder.parse(xmlFile);

        Element root = xmlDoc.getDocumentElement();
        NodeList rootChildNodes = root.getChildNodes();
```

eXtensible Markup Language (XML)

XML Parsers

XML DOM Parser in Java

```
String gender=root.getElementsByTagName("gender").item(0).getTextContent();

    int noRootChildNodes = rootChildNodes.getLength();

    for(int i=0;i<noRootChildNodes;i++)  {
            Node node = rootChildNodes.item(i);

            if(node.getNodeName().equals("name"))  {
                    if(gender.equals("Male"))
                            System.out.println("Name: Mr. "+node.getTextContent());
                    else
                            System.out.println("Name: Ms. "+node.getTextContent());
            }
```

eXtensible Markup Language (XML)

XML Parsers

XML DOM Parser in Java

```
if(node.getNodeName().equals("email"))
        System.out.println("Email: "+node.getTextContent());

if(node.getNodeName().equals("phone")) {
        String couCode =node.getAttributes().item(0).getTextContent();
        System.out.println("Phone: "+couCode+" "+node.getTextContent());
}

if(node.getNodeName().equals("gender"))
        System.out.println("Gender: "+node.getTextContent());

if(node.getNodeName().equals("birthday"))
        System.out.println("Birthday: "+node.getTextContent());
}}}
```

eXtensible Markup Language (XML)

XML Processors

Example:

XML File:

```
<?xml version="1.0" encoding = "UTF-8"?>
<address>
<name>Samar Singh</name>
<email>samar.singh@XXX.com</email>
<phone countryCode="+91">99XX556622</phone>
<gender>Male</gender>
<birthday>22-03-1985</birthday>
</address>
```

Program Output:

```
Name: Mr. Samar Singh
Email: samar.singh@XXX.com
Phone: +91 99XX556622
Gender: Male
Birthday: 22-03-1985
```

eXtensible Markup Language (XML)

XML Parsers

XML DOM Parser

Advantages:

- Easy to navigate and manipulate the XML structure.
- Random access to any part of the document.

Disadvantages:

- Requires a large amount of memory as it loads the entire XML document into memory.
- Slow parsing speed because it requires more memory and time as the entire document must be parsed and stored before any operations can be performed.

eXtensible Markup Language (XML)

XML Parsers

XML SAX Parser

"XML SAX Parser is a tool that reads XML documents in a streaming fashion, triggering events as it encounters different elements, attributes, and text, without loading the entire document into memory."

- Event-Driven Processing: The SAX parser processes XML data sequentially and generates events (like start and end of elements) that can be handled by user-defined callbacks, allowing for real-time processing of the document.

- No In-memory Representation: Since the SAX parser does not construct a tree structure of the entire document, it does not consume memory for storing the whole XML document.

- Suitable for Large XML Files: SAX can handle large XML files where only a subset of the document needs to be processed at any given time, allowing for efficient memory usage and faster processing.

eXtensible Markup Language (XML)

XML Parsers

XML SAX Parser in Java

```
import java.io.File;
import java.io.IOException;
import javax.xml.parsers.*;

public class SAXDemo {

   public static void main(String[] args) throws Exception  {
      SAXParserFactory factory = SAXParserFactory.newInstance();
      factory.setNamespaceAware(true);
      SAXParser saxParser = factory.newSAXParser();
      saxParser.parse(new File("address2.xml"), new AddressHandler());
   }
}
```

eXtensible Markup Language (XML)

XML Parsers

XML SAX Parser in Java

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class AddressHandler extends DefaultHandler {
    String tagName=null;
    Attributes attributes=null;
    String name=null;
    String email=null;
    String phone=null;
    String gender=null;
    String birthday=null;
    String title = "Mr.";  // Default title is "Mr." unless changed by gender
```

eXtensible Markup Language (XML)

XML Parsers

XML SAX Parser in Java

```
@Override
public void startElement(String uri, String localName, String qName, Attributes attributes) throws
SAXException {
    tagName = qName;
    this.attributes = attributes;
}
```

eXtensible Markup Language (XML)

XML Parsers

XML SAX Parser in Java

```
@Override
    public void endElement(String uri, String localName, String qName) throws SAXException {
      if(qName.equals("address"))
      {
        if (gender.equalsIgnoreCase("Male")) {
            title = "Mr.";
        } else if (gender.equalsIgnoreCase("Female")) {
            title = "Ms.";
        }
        System.out.println("Name: "+title+" "+name);
        System.out.println("Email: "+email);
        System.out.println("Phone: "+phone);
        System.out.println("Gender: "+gender);
        System.out.println("Birthday: "+birthday);        }            }
```

eXtensible Markup Language (XML)

XML Parsers

XML SAX Parser in Java

```
@Override
    public void characters(char[] ch, int start, int length) throws SAXException {
        super.characters(ch, start, length);
        String tagText = new String(ch,start,length);
        if(tagText.trim().length()>0)
        {
         if (tagName.equals("name"))          {      name = tagText;      }
          else if (tagName.equals("email"))   {      email = tagText;     }
          else if (tagName.equals("phone"))   {      phone = tagText;     }
          else if (tagName.equals("gender"))  {      gender = tagText;    }
          else if (tagName.equals("birthday")) {     birthday = tagText;  }
        }
    }
}
```

eXtensible Markup Language (XML)

XML Processors

Example:

XML File:

```
<?xml version="1.0" encoding = "UTF-8"?>
<address>
<name>Samar Singh</name>
<email>samar.singh@XXX.com</email>
<phone countryCode="+91">99XX556622</phone>
<gender>Male</gender>
<birthday>22-03-1985</birthday>
</address>
```

Program Output:

```
Name: Mr. Samar Singh
Email: samar.singh@XXX.com
Phone: +91 99XX556622
Gender: Male
Birthday: 22-03-1985
```

eXtensible Markup Language (XML)

XML Parsers

XML SAX Parser

Advantages:

- Low memory usage as it doesn't load the entire document into memory.
- Faster parsing speed compared to DOM.

Disadvantages:

- No random access to the XML document's parts.
- More complex code as the developer has to handle events..