Unit-1 Web Technology Question Bank -BCS-502

1. Write the steps for planning and publishing your website. Discuss the basic elements of good web site.

Steps for Planning and Publishing a Website

1. Define Your Goals and Purpose:
- Identify the primary purpose of the website (e.g., informational, e-commerce, portfolio).
- Define the target audience and their needs.

2. Research and Plan Content:
- Organize content into clear sections (e.g., Home, About, Services, Contact).
- Draft text, gather images, videos, and other media.

3. Choose a Domain Name:
- Select a domain name that reflects your brand and is easy to remember.
- Register the domain through a domain registrar like GoDaddy or Namecheap.

4. Select a Web Hosting Service:
- Choose a hosting provider based on your needs (shared hosting, VPS, or dedicated server).
- Ensure the hosting provider offers good uptime, customer support, and scalability.

5. Design the Website:
- Decide between using a website builder (e.g., Wix, Squarespace) or custom development.
- Create wireframes or mockups to visualize the layout.
- Use responsive design to ensure compatibility with devices of all screen sizes.

6. Develop the Website:
- Use web technologies like HTML, CSS, and JavaScript or frameworks like React or Angular.
- Implement a Content Management System (CMS) like WordPress if required.
- Add navigation menus, forms, and interactive elements.

7. Optimize for Search Engines (SEO):
- Use relevant keywords in titles, headings, and content.
- Optimize images and meta tags.
- Ensure fast page loading times.

8. Test the Website:
- Test across multiple devices and browsers for compatibility.
- Check all links, forms, and media for functionality.
- Ensure security measures like HTTPS are in place.

9. Publish the Website:
- Upload files to the hosting server using FTP or a CMS.
- Link the domain name to the hosting server.
- Perform a final check before going live.

10. Maintain and Update:
- Regularly update content and software to keep the website fresh and secure.
- Monitor analytics to improve user experience and performance.

Basic Elements of a Good Website

1. User-Centered Design:
- Intuitive navigation with clear menus and links.
- Responsive design for mobile, tablet, and desktop compatibility.

	2. Visual Appeal: • Use a clean and professional layout. • Maintain a consistent color scheme and typography. • Use high-quality images and graphics. **3. Content Quality:** • Provide clear, concise, and engaging text. • Organize information logically with headings and subheadings. • Update content regularly to maintain relevance. **4. Fast Loading Speed:** • Optimize images and minimize scripts to reduce load times. • Use caching and Content Delivery Networks (CDNs). **5. SEO Optimization:** • Use appropriate meta tags and descriptions. • Incorporate keywords naturally into content. • Create an XML sitemap and submit it to search engines. **6. Security:** • Use HTTPS for encrypted communication. • Protect user data with secure forms and anti-spam measures. • Regularly update software and plugins to fix vulnerabilities. **7. Accessibility:** • Follow web accessibility guidelines (e.g., WCAG) to accommodate all users. • Use alt text for images and provide keyboard navigation. **8. Clear Call-to-Action (CTA):** • Include prominent CTAs to guide users (e.g., "Sign Up," "Buy Now," "Contact Us"). **9. Contact Information:** • Provide clear and easy-to-find contact details. • Include forms, email addresses, or chat options for user interaction. **10. Analytics and Feedback:** • Integrate tools like Google Analytics to monitor traffic and user behavior. • Collect user feedback to improve the website.
2.	Why early planning is useful to develop an effective website. Give proper example in Favor of your reason. **Reasons for Early Planning** **1. Clarity of Purpose and Goals** • Early planning helps define the purpose of the website (e.g., informational, e-commerce, portfolio). • **Example:** A non-profit organization decides to launch a donation website. By planning early, they identify "donation simplicity" as a primary goal, which ensures the site has streamlined donation forms and clear CTAs for potential donors. **2. Identifying Target Audience** • Knowing your audience influences design, content, and functionality. • **Example:** A fashion retailer targeting millennials might focus on mobile-first design and integrate social media features, identified during the planning phase. **3. Efficient Resource Allocation** • Planning helps allocate time, budget, and technical resources effectively. • **Example:** A start-up with limited resources can prioritize key features, such as a product showcase and online payments, ensuring a functional site is delivered within budget.

	4. Reducing Scope Creep • Planning defines project scope and prevents unplanned changes. • **Example:** During planning, an e-commerce store defines its features (product search, cart, checkout). This clarity avoids last-minute additions like a loyalty program, which could delay the launch. **5. Ensuring a Better User Experience** • Planning considers navigation, responsive design, and accessibility from the outset. • **Example:** A news website plans a simple, intuitive layout during the early stages, ensuring readers can easily find articles and navigate categories. **6. SEO and Content Strategy** • Early planning aligns content creation and SEO, ensuring optimized content at launch. • **Example:** A travel blog identifies popular keywords and plans articles around them, helping it rank well in search engines from day one. **7. Risk Mitigation** • Identifying potential risks (technical challenges, budget overruns) early avoids costly setbacks. • **Example:** A university planning a student portal recognizes the need for robust security features during the planning phase, preventing data breaches. **8. Time and Cost Efficiency** • A well-planned project minimizes revisions, saving time and costs. • **Example:** An app developer plans a minimum viable product (MVP) for a service website, ensuring the launch happens on schedule while leaving room for future updates.
3.	Describe the various web development strategies. Web Development Strategies Developing a website involves a well-structured approach to ensure it is responsive, user-friendly, and meets the needs of its audience. Here are various web development strategies that play a crucial role in the success of a website: 1. Define the Purpose and Objectives Clearly outline the website's purpose (e.g., e-commerce, blogging, portfolio, informational). Set measurable objectives such as increasing traffic, boosting sales, or enhancing user engagement. Example: An e-commerce website may aim to increase online sales by 20% within six months. 2. Understand the Target Audience Identify the demographics, preferences, and behaviors of the target audience. Tailor the website design, content, and functionality to meet their needs. Example: A website targeting millennials may focus on mobile-first design and integration with social media. 3. Plan the Website Structure Use a site map to organize content and navigation. Focus on intuitive navigation to improve user experience (UX). Ensure critical pages (e.g., home, contact, about) are easily accessible. 4. Prioritize Responsive Design Ensure the website is fully responsive and adapts to different devices (desktop, tablet, mobile). Use frameworks like Bootstrap or Foundation for consistent responsiveness.

	Optimize layouts for various screen sizes. 5. Optimize Performance Minimize load times using techniques like image optimization, caching, and Content Delivery Networks (CDNs). Ensure the website scores well in performance tools like Google PageSpeed Insights. Example: Compressing images and using lazy loading to reduce bandwidth usage. 6. Emphasize Security Use HTTPS to encrypt data and secure the website. Implement measures like firewalls, regular updates, and secure authentication protocols. Example: An e-commerce site should ensure secure payment processing through SSL certificates. 7. Choose the Right Technology Stack Select appropriate front-end (HTML, CSS, JavaScript frameworks) and back-end technologies (Node.js, PHP, Python). Use Content Management Systems (CMS) like WordPress, Drupal, or custom solutions for dynamic content. 8. Focus on SEO (Search Engine Optimization) Optimize content with relevant keywords. Use meta tags, alt text for images, and structured data for better search engine visibility. Ensure the website has a clean URL structure and a sitemap. 9. Implement Accessibility Standards Ensure the website adheres to WCAG (Web Content Accessibility Guidelines). Make content accessible for people with disabilities, such as screen readers for visually impaired users. Example: Use proper ARIA (Accessible Rich Internet Applications) labels for elements. 10. Integrate Analytics Tools Use tools like Google Analytics or Hotjar to monitor user behavior. Track metrics such as page views, bounce rate, and user flow to identify areas for improvement. 11. Content Strategy Develop engaging and relevant content tailored to the audience. Update the website regularly with fresh content to maintain relevance. Example: A blog should consistently publish new posts on trending topics in its niche. 12. Leverage Social Media and Marketing Integrate social media sharing buttons for content. Use social media platforms to drive traffic and promote the website. Example: Launch targeted ad campaigns on Facebook and Instagram. 13. Test and Optimize Conduct usability testing to identify and resolve issues before launch. Use A/B testing to optimize design and content for conversions. Test across multiple browsers and devices. 14. Maintenance and Updates Regularly update the website to fix bugs, improve features, and add new content. Monitor for downtime and resolve technical issues promptly. Example: An online store may need to frequently update product listings and prices. 15. Scalability Planning Design the website to handle increased traffic as it grows. Use cloud services like AWS or Azure for scalable hosting solutions. Example: A start-up site should be ready to accommodate more users as its audience grows.

4. Describe the objective of any website, which type of essential skill is required being a member of web project team.

Objectives of a Website

The objective of a website varies depending on its purpose, audience, and industry. Common objectives include:

1. **Information Sharing**
   - To provide users with accurate, relevant, and up-to-date information.
   - Example: News portals, educational websites.
2. **Brand Promotion**
   - To establish an online presence and promote a business, product, or service.
   - Example: Corporate websites, portfolio sites.
3. **E-commerce**
   - To sell products or services directly to customers.
   - Example: Amazon, eBay.
4. **Community Building**
   - To create a platform for user interaction and engagement.
   - Example: Social media sites, forums.
5. **Lead Generation**
   - To attract and capture leads for conversion into customers.
   - Example: Landing pages for digital marketing campaigns.
6. **Customer Support**
   - To assist users by providing FAQs, live chat, or contact forms.
   - Example: Helpdesk websites, support portals.

Essential Skills for a Web Project Team

Creating a successful website requires a collaborative effort from a team with diverse skills. Below are the key roles and essential skills required:

---

1. Web Designer

- **Skills Required:**
  - Proficiency in design tools (e.g., Adobe XD, Figma, Sketch).
  - Knowledge of HTML, CSS, and JavaScript for designing responsive layouts.
  - Creativity and understanding of UI/UX principles.
- **Role:**
  - Design the visual layout, color schemes, typography, and overall aesthetics.

---

2. Web Developer

- **Skills Required:**
  - Front-End: HTML, CSS, JavaScript, frameworks like React, Angular, or Vue.js.
  - Back-End: Knowledge of server-side languages (e.g., PHP, Python, Ruby, Node.js).
  - Database Management: SQL, MongoDB, or similar.
  - Version Control: Git.
- **Role:**
  - Build and maintain the website's functionality and integrate front-end with back-end.

---

3. Content Creator/Writer

- **Skills Required:**
  - Excellent writing and editing skills.
  - Knowledge of SEO (Search Engine Optimization).
  - Ability to create engaging and audience-relevant content.
- **Role:**
  - Develop text, images, and videos to populate the website.

---

4. SEO Specialist

- **Skills Required:**
  - Understanding of search engine algorithms.
  - Experience with tools like Google Analytics, SEMrush, or Ahrefs.
  - Ability to optimize meta tags, keywords, and content for search engines.
- **Role:**
  - Increase website visibility and improve search engine ranking.

---

5. Project Manager

- **Skills Required:**
  - Strong organizational and communication skills.
  - Familiarity with project management tools (e.g., Trello, Asana, Jira).
  - Ability to coordinate between team members and stakeholders.
- **Role:**
  - Plan, execute, and oversee the web project to meet deadlines and objectives.

---

6. Quality Assurance (QA) Tester

- **Skills Required:**
  - Attention to detail.
  - Knowledge of testing tools and methodologies.
  - Understanding of usability and browser compatibility testing.
- **Role:**
  - Ensure the website is free from bugs, functions correctly, and is user-friendly.

---

7. Digital Marketer

- **Skills Required:**
  - Knowledge of social media platforms, email marketing, and PPC campaigns.
  - Experience with analytics tools to measure performance.
  - Creativity in planning and executing marketing strategies.
- **Role:**
  - Promote the website and drive traffic through various digital channels.

---

8. Security Specialist

- **Skills Required:**
  - Expertise in web security protocols.
  - Knowledge of firewalls, encryption, and secure coding practices.
  - Ability to identify vulnerabilities and mitigate risks.
- **Role:**
  - Protect the website from cyber threats and ensure data privacy.

5. Explain the difference between web design and web development. Explain the various web development phases.

Difference Between Web Design and Web Development

Aspect	Web Design	Web Development
Definition	Focuses on the visual aesthetics and user experience of a website.	Focuses on the technical implementation and functionality of the website.
Primary Goal	To create an appealing and user-friendly interface.	To build and maintain the website's core structure and functionality.
Skills Involved	UI/UX design, graphic design, knowledge of design tools (e.g., Figma, Photoshop), and basic HTML/CSS.	Programming languages (HTML, CSS, JavaScript, Python, PHP), databases, and server-side scripting.
Tools	Design software like Adobe XD, Figma, or Sketch.	Development environments like VS Code, Sublime Text, and frameworks like React, Angular, or Laravel.
Output	Produces wireframes, prototypes, and style guides.	Produces the fully functional website or application.
User Interaction	Directly impacts how the user perceives the website.	Impacts the functionality and interactivity of the website.
Example Task	Designing the layout of a homepage.	Creating a user login system or database integration.

Phases of Web Development
The web development process is divided into several key phases to ensure a structured and efficient workflow:

---

1. Planning Phase

- **Objective:** Define the purpose, goals, and requirements of the website.
- **Key Activities:**
  - Identifying the target audience.
  - Creating a project plan and timeline.
  - Outlining the website structure using a sitemap.
- **Deliverables:**
  - Project documentation.
  - Sitemap or site architecture.
- **Example:** Planning an e-commerce website with product categories, a shopping cart, and a payment gateway.

---

2. Design Phase

- **Objective:** Create the visual layout and design of the website.
- **Key Activities:**
  - Developing wireframes and prototypes.
  - Choosing typography, color schemes, and imagery.
  - Designing user interfaces (UI) and ensuring an optimal user experience (UX).
- **Deliverables:**

	- Wireframes and mockups. - Style guides or design specifications. **Example:** Designing a responsive homepage with an intuitive navigation bar. <hr>3. Development Phase **Objective:** Translate designs into a functional website or application. **Key Activities:** - Writing code for the front-end (HTML, CSS, JavaScript). - Implementing back-end functionality (server-side scripting, databases). - Integrating third-party APIs or services. **Deliverables:** - Fully functional website or application. **Example:** Developing a login system using PHP and MySQL. <hr>4. Testing Phase **Objective:** Ensure the website is free from errors and performs as expected. **Key Activities:** - Testing for bugs, broken links, and compatibility across browsers and devices. - Conducting performance tests to optimize loading speed. - Checking for security vulnerabilities. **Deliverables:** - Bug-free, fully functional website. **Example:** Testing a shopping cart's checkout process to ensure smooth transactions. <hr>5. Deployment Phase **Objective:** Launch the website and make it available to the target audience. **Key Activities:** - Setting up hosting and domain. - Uploading files to the server. - Performing a final review before going live. **Deliverables:** - Live website accessible on the internet. **Example:** Deploying a blog site on a hosting platform like AWS or Bluehost. <hr>6. Maintenance Phase **Objective:** Ensure the website remains functional, up-to-date, and secure. **Key Activities:** - Regularly updating content and software. - Monitoring site performance and fixing issues promptly. - Enhancing features based on user feedback. **Deliverables:** - Updated and secure website. **Example:** Updating product prices and adding new features to an e-commerce site.
6.	Differentiate Between HTML and XML. What is XML schema? Compare XML Schema and XML DTD with neat Example. **The main difference between XML and HTML:** HTML is an abbreviation for HyperText Markup Language while XML stands for eXtensible Markup Language.The differences are as follows:-

	1. HTML was designed to display data with focus on how data looks while XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is. 2. HTML is a markup language itself while XML provides a framework for defining markup languages. 3. HTML is a presentation language while XML is neither a programming language nor a presentation language. 4. HTML is case insensitive while XML is case sensitive. 5. HTML is used for designing a web-page to be rendered on the client side while XML is used basically to transport data between the application and the database. 6. HTML has its own predefined tags while what makes XML flexible is that custom tags can be defined and the tags are invented by the author of the XML document. 7. HTML is not strict if the user does not use the closing tags but XML makes it mandatory for the user the close each tag that has been used. 8. HTML does not preserve white space while XML does. 9. HTML is about displaying data, hence static but XML is about carrying information, hence dynamic. Thus,it can be said that HTML and XML are not competitors but rather complement to each other and clearly serving altogether different purposes.
7.	What is the difference between external and internal DTD? Create XML file explains the attributes of phones using DTD. 1) Internal / Embedded DTD. <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE student [<!ELEMENT student (id,name,age,addr,email,ph)> <!ELEMENT id (#PCDATA)> <!ELEMENT name (#PCDATA)> <!ELEMENT age (#PCDATA)> <!ELEMENT addr (#PCDATA)> <!ELEMENT email (#PCDATA)> <!ELEMENT ph (#PCDATA)>]> <student> <id>543</id> <name>Ravi</name> <age>21</age> <addr>Guntur</addr> <email>nsr@gmail.com</email> <ph>9855555</ph> <gender>male</gender> </student> 2) External DTD. <!ELEMENT student (id,name,age,addr,email)> <!ELEMENT id (#PCDATA)> <!ELEMENT name (#PCDATA)> <!ELEMENT age (#PCDATA)> <!ELEMENT addr (#PCDATA)> <!ELEMENT email (#PCDATA)> Save the above code as “student.dtd” and prepare “student.xml” as follows...

	<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE student SYSTEM "student.dtd"> <student> <id>543</id> <name>Ravi</name> <age>21</age> <addr>Guntur</addr> <email>nsr@gmail.com</email> </student> In the above example we are using <!DOCTYPE student SYSTEM "student.dtd"> which is used to provide “student.dtd” code in our “student.xml” file. If the above xml code follows the exact rules defined in DTD then we can conclude that our xml document is a valid document. Otherwise it is an invalid document. **XML Document: phones.xml** xml Copy code <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE phones SYSTEM "phones.dtd"> <phones> <phone brand="Apple" model="iPhone 14" price="999.99" os="iOS" ram="6GB" storage="128GB"/> <phone brand="Samsung" model="Galaxy S23" price="849.99" os="Android" ram="8GB" storage="256GB"/> <phone brand="Google" model="Pixel 8" price="799.99" os="Android" ram="8GB" storage="128GB"/> <phone brand="OnePlus" model="11 Pro" price="699.99" os="Android" ram="12GB" storage="256GB"/> <phone brand="Xiaomi" model="Mi 13" price="499.99" os="Android" ram="8GB" storage="128GB"/> </phones>
8.	Discuss DTD. How it is different from XSD. Demonstrate to create a XML document of 10 students of Third year, Add their roll no , marks obtained in 5 subjects , total marks and percentage and validate using DTD. XML DTD: (Document Type Definition) An XML document with correct syntax is called "Well Formed".□ An XML document validated against a DTD is both "Well Formed" and "Valid".□ A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a DTD.□ DTD is the basic building block of XML. The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements. □---------- <!DOCTYPE book

[
<!ELEMENT book
(title,author,price)> <!ELEMENT
title (#PCDATA)> <!ELEMENT author
(#PCDATA)> <!ELEMENT price
(#PCDATA)>]> ----------
The DTD above is interpreted like this:
!DOCTYPE book defines that the root element of the document is book□

!ELEMENT book defines that the book element must contain the elements:□
"title, author, price"
!ELEMENT title defines the title element to be of type "#PCDATA"□
!ELEMENT author defines the author element to be of type "#PCDATA"□
!ELEMENT price defines the price element to be of type "#PCDATA"□
Note: PCDATA: Parse able Character Data, CDATA: Character Data.
There are two types of DTDs:
1) Internal / Embedded DTD.
2) External DTD.

Difference Between DTD (Document Type Definition) and XSD (XML Schema Definition)

Aspect	**DTD (Document Type Definition)**	**XSD (XML Schema Definition)**
Purpose	Defines the structure and legal elements of an XML document, but with limited validation capabilities.	Defines the structure, content, and data types of XML documents, with extensive validation support.
Syntax	Uses a simplified, older syntax that is less flexible.	Uses XML syntax itself, making it more extensible and standard for XML documents.
Validation	Provides basic validation to ensure that the document follows the correct structure.	Provides detailed validation, including data types (e.g., integer, string), constraints, and even custom validation rules.
Support for Data Types	Does not support data types (all data is treated as text).	Supports built-in and custom data types (e.g., integer, decimal, date, etc.).
Namespaces	Does not support XML namespaces, leading to potential name conflicts.	Fully supports XML namespaces, preventing conflicts and improving scalability.
Reusability	Limited reusability. You cannot define reusable data types or complex structures.	Highly reusable with the ability to define complex types, groups, and elements.

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE students SYSTEM "students.dtd">
<students>
  <student>

	<rollNo>101</rollNo> <name>John Doe</name> <marks> <subject>80</subject> <subject>75</subject> <subject>85</subject> <subject>90</subject> <subject>88</subject> </marks> <totalMarks>418</totalMarks> <percentage>83.6</percentage> </student> <student> <rollNo>102</rollNo> <name>Jane Smith</name> <marks> <subject>78</subject> <subject>84</subject> <subject>80</subject> <subject>82</subject> <subject>79</subject> </marks> <totalMarks>403</totalMarks> <percentage>80.6</percentage> </student> <!-- Add 8 more students similarly --> </students>
9.	Explain DOM. What are the XML Parsers? Explain different type of parsers with their advantage and disadvantage. **DOM (Document Object Model)** The **Document Object Model (DOM)** is a programming interface for XML and HTML documents. It represents the document as a tree structure, where each node corresponds to a part of the document (e.g., elements, attributes, text). DOM allows developers to manipulate the structure, style, and content of documents dynamically using scripting languages like JavaScript or Python. **Key Features of DOM** 1. **Tree Structure:** The document is modeled as a tree of objects, with the root node representing the document. 2. **Dynamic Updates:** DOM enables reading and updating the document's structure and content dynamically. 3. **Cross-Platform:** It is platform- and language-independent. **Example: XML Document** xml Copy code <bookstore> <book category="fiction"> <title lang="en">The Great Gatsby</title> <author>F. Scott Fitzgerald</author>

<price>10.99</price>
</book>
</bookstore>

DOM Representation:

- The <bookstore> is the root element.
- <book> is a child of <bookstore> and has an attribute category.
- <title>, <author>, and <price> are children of <book>.

XML Parsers

An **XML Parser** is a software library or tool used to read, manipulate, and validate XML documents. Parsers process the XML document and transform it into a format suitable for manipulation by a programming language.

Types of XML Parsers

1. **DOM Parser**
2. **SAX Parser**
3. **StAX Parser**

---

1. DOM Parser

Description:

- Reads the entire XML document and loads it into memory as a tree structure.
- Provides a random-access interface for traversing and manipulating the document.

Advantages:

- Easy to use and navigate the XML structure.
- Suitable for small to medium-sized XML documents.
- Allows dynamic updates to the document.

Disadvantages:

- Consumes a lot of memory, as the entire document is loaded.
- Slower for large XML documents.

Use Case:

Ideal for applications where the XML document is small and frequent read/write operations are required.

---

2. SAX Parser (Simple API for XML)

Description:

- Processes the XML document sequentially, element by element, using event-based callbacks.
- Does not load the entire document into memory.

Advantages:

- Memory efficient, as it processes data one piece at a time.
- Faster for large XML documents.

Disadvantages:

- Does not allow random access; processing is strictly sequential.
- More complex to implement, as you need to handle events manually.
- Cannot modify the XML document.

Use Case:

Best for reading large XML files where only specific parts of the document are needed.

---

3. StAX Parser (Streaming API for XML)

Description:

- A pull-parsing model where the application "pulls" data from the parser as needed.
- Processes XML documents sequentially like SAX but gives more control to the programmer.

Advantages:

- Memory efficient, as it processes data on demand.
- Provides more control over parsing compared to SAX.

Disadvantages:

- More complex to implement than DOM.
- Limited ability to modify XML documents compared to DOM.

Use Case:

Suitable for applications requiring selective and sequential parsing, such as large data streams.

Comparison of XML Parsers

Aspect	DOM Parser	SAX Parser	StAX Parser
Parsing Method	Loads entire document into memory.	Event-driven, sequential parsing.	Pull-based, sequential parsing.
Memory Usage	High, as the entire document is loaded.	Low, as it processes data piece by piece.	Low, processes data on demand.
Access Method	Random access to any part of the document.	Sequential access only.	Sequential access with pull control.
Complexity	Easy to use.	Complex to implement event handling.	Moderate complexity.
Modification	Allows modification.	Does not allow modification.	Limited modification.
Use Case	Small to medium XML files.	Large XML files with simple processing needs.	Large XML files with selective processing.

10. Differentiate between DOM and SAX.

Difference between SAX and DOM parsers.

DOM	SAX
Tree model parser (Tree of nodes)	Event based parser (Sequence of events)
DOM loads the file into the memory and then parse the file	SAX parses the file at it reads i.e. Parses node by node
Has memory constraints since it loads the whole XML file before parsing	No memory constraints as it does not store the XML content in the memory
DOM is read and write (can insert or delete the node)	SAX is read only i.e. can't insert or delete the node
If the XML content is small then prefer DOM parser	Use SAX parser when memory content is large
Backward and forward search is possible for searching the tags and evaluation of the information inside the tags. So this gives the ease of navigation	SAX reads the XML file from top to bottom and backward navigation is not possible
Slower at runtime	Faster at runtime