ASSIGNMENT - 1

1. How internet works?

The internet connects devices globally through unique IP addresses, routing data packets via routers and switches. Protocols like TCP/IP manage data transmission, ensuring reliable delivery. Severs host content and services, responding to user requests with data packets .DNS translates domains names to IP addresses, enabling easy access. Encryption secures data privacy during transmission, facilitating seamless global communication and resource access.

2. How browser works?

A browser retrieves web pages by sending requests to servers via the internet. It interprets HTML to structure the page, CSS for styling, and JavaScript for dynamic behavior. The browser's rendering engine processes these elements, into a visual representation. It manages caching to store resources locally for faster loading on revisits. Browsers support plugins/extensions for added functionality. They maintain a browsing history and session data, and ensure security through sandboxing and HTTPS protocols. Finally, browsers allow users to interact with web content via a graphical user interface.

3. What is server?

A server is a computer or software that provides resources, data, or services to other computers, known as clients, over a network. It responds to requests from clients, such as web browsers or applications, by delivering data

or performing tasks. Servers can host websites, handle email, store files, manage databases, and more, depending on their configuration and purpose.

4. What are the types of server available?

There are several types of servers available, each serving specific purpose.

- Web Server: Delivers web pages and content to clients.
- File Server: Stores and manage files accessible to clients over a network.
- Database Server: Manages and provides access to databases and their data.
- Application Server: Executes applications and provides middleware services.
- Mail Server: manages email communication, storage, and routing.
- Proxy Server: Acts as an intermediary between clients and other servers, providing various functionalities such as caching and security.
- Virtual Server: Runs multiple virtual environments on a single physical server.
- FTP Server: Handles file transfers over FTP protocols.
- DNS Server: revolves domain names to IP addresses.
- Print Server: Manages printers and point jobs on a network.

5. What is SEO? Importance of SEO?

Search Engine Optimization (SEO) refers to the practice of enhancing a website to improve its visibility for relevant searches on search engines like Google, Bing, and yahoo.

Importance of SEO :

- Increased traffic: organic search is a significant source of website traffic.
- Cost-Effective: It focuses on organic traffic, which does not require ongoing payment.
- User Experience: SEO practices improve the usability and user experience of a website.
- Brand Credibility and Trust: Good SEO helps establish authority and brand awareness.
- Competitive Advantage: A well-optimized website can outrank competitors in search results, capturing more market share.
- Local SEO: Local SEO is crucial for businesses that operate on a regional level.
- Insights and Analytics: SEO tools provide valuable insights into customer behavior, preference, and trends.
- Long-Term strategy: SEO is a long-term strategy that continues to provide benefits over time.

6. What is Accessibility?

Accessibility refers to the practice of designing and developing websites, applications, and digital content so that they can be used by everyone, including people with disabilities. This involves ensuring that content is perceivable, operable, understandable, and robust for all users, including those using assistive technologies like screen readers.

7. What is markup Language?

A markup language is a system for annotating a document in a way that is syntactically distinguishable from the text, indicating how the document should

be structured or formatted. Markup languages are used to describe the layout, formatting, and sometimes the semantics of text and other data. Example: HTML, XML, markdown. Markup language provide a way to separate the content from its presentation.

8. What is HTML?

HTML, which stands for HyperText markup Language, is the standard markup language used to create and design documents on the World Wide Web. It defines the structure and layout of a web page by using a variety of tags and attributes to specify how content should be displayed on browsers.

HTML forms the backbone of every web page on the internet, providing the basic structure and semantics that browsers use to render content. It works in conjunction with Cascading Style Sheets(CSS) for styling and JavaScript interactive behavior to create dynamic and visually appealing web pages.

9. What is browser Engine?

A browser engine, also known as a layout engine or rendering engine, is a core component of web browsers that interprets and displays web content. It is responsible for rendering HTML, CSS, and JavaScript code into visual elements on the screen that users interact with.

Examples of Browser Engine:

- Blink (used in chrome and engine).
- WebKit (used in Safari).
- Gecko (used in Firefox).

10. What is rendering engine? Share the available rendering engine?

A rendering engine is a core component of a web browser that interprets HTML and CSS code to display web pages. It takes the structured data from web pages and converts it into the visual layout users see on their screens.

Here are some widely used rendering engines:

- Blink: Used by Google chrome, Microsoft Edge, Opera.
- WebKit: Used by: Apple Safari, formerly Google Chrome (until Chrome 27).
- Gecko: Used by: Mozilla Firefox.

11. What is JavaScript engine? Share the available JS engine? Purpose of JDS engine?

A JavaScript engine is a computer program that executes JavaScript code. It essentially translates JavaScript code into machine code that the computer's processor can execute directly.

JavaScript engines are a crucial component of web browsers (like Chrome, Firefox, Safari, Edge) and server-side environments (like Node.js). They are responsible for making JavaScript a versatile and widely used language for both client-side and server-side development. Each browser or environment may use a different JavaScript engine, tailored for its specific requirements and performance goals.

Here are some of the prominent JavaScript engines used today:

1. V8:

- Developed by Google for the Chrome browser and Node.js.
- Known for its speed and efficiency, uses just-in-time (JIT) compilation.
- Written in C++.

2. SpiderMonkey :
   - Developed by Mozilla, used in Firefox. ○ One of the oldest JavaScript engines, originally created by Brendan Eich.
   - Supports JIT compilation.
3. JavaScriptCore (Nitro):
   - Developed by Apple, used in Safari and WebKit.
   - Designed to be lightweight and fast. ○ Used in various Apple platforms including macOS, iOS, and watchOS.
4. Chakra (Legacy):
   ○ Developed by Microsoft, initially used in Internet Explorer and later in Microsoft Edge (replaced by V8 in newer versions of Edge). ○ Designed for efficient performance and compatibility with Microsoft's platforms.
5. SpiderMonkey (Rhino):
   ○ Another version of SpiderMonkey, originally developed by Netscape and later maintained by Mozilla.
   ○ Written in Java and used primarily for embedding JavaScript in Java applications.
6. JerryScript:
   ○ Developed by Samsung, optimized for resource-constrained devices and IOT applications.

   ○ Designed to be lightweight and efficient, suitable for microcontrollers and embedded systems.

The purpose of a JavaScript engine is to execute JavaScript code.

Main functions and goals of a JavaScript engine:

*Parsing: The engine reads and parses JavaScript code to understand its structure and syntax.

* Compilation: The engine translates JavaScript code into machine understandable code, often in the form of bytecode or machine code. This process may involve optimizations to improve execution speed.

*Optimization: Modern JavaScript engines employ sophisticated optimization techniques to enhance performance. This includes dynamic optimization based on runtime profiling of code execution patterns.

*Execution: Once compiled and optimized, the engine executes the JavaScript code, performing computations, manipulating objects, and interacting with the environment (e.g., DOM manipulation in browsers).

*Memory Management: JavaScript engines manage memory allocation and deallocation to ensure efficient use of system resources and prevent memory leaks.

* Error Handling: They handle errors that occur during execution, providing useful debugging information to developers.

12. How Website works?

Websites work by storing files on web servers, which are special computers connected to the internet. When someone types a website address (like www.example.com) into their browser, the browser sends a request to the

web server. The server then sends the website's files back to the browser, which interprets them and displays the site's content and design to the user. This process involves technologies like HTML, CSS, JavaScript, and servers that work together to create the web pages we see and interact with online.

13. What is Data structure?

A data structure is a way of organizing and storing data in a computer so that it can be accessed and used efficiently. It defines the relationship between the data elements, how they are organized, and the operations that can be performed on them. Essentially, data structures provide a means to manage and manipulate data effectively.

Common types of data structures include:

- Arrays
- Linked List
- Stack
- Queue
- Trees
- graphs

14. Explain tree data structure?

A tree data structure is a hierarchical structure composed of nodes, where each node can have zero or more child nodes, but only one parent node (except for the root node, which has no parent). It is a widely used abstract data type that mimics the structure of a real-world tree, with nodes representing entities connected by edges (branches).

Trees can be balanced or unbalanced, depending on how evenly the nodes are distributed across levels, affecting the performance of operations such as insertion, deletion, and search. They are fundamental in computer science and are implemented in various programming languages through classes and pointers or references.

15. What is user Agent? Share the list and its purpose?

A user agent refers to a string of information that identifies the browser and operating system of a user accessing the internet. It is transmitted by the browser to websites and web servers during HTTP requests.

User agents are essential for web servers to provide content that is optimized for the user's browser and operating system. They can also help in tracking usage statistics and trends across different platforms. However, user agent strings can also be manipulated or spoofed, which can affect how accurately websites can determine the characteristics of the user's browser and device.

Purpose of User-Agent**:**

1. Content Negotiation: Servers use the UA to determine how to best render content for the requesting client. For example, a mobile site might serve a different layout optimized for smaller screens.
2. Browser Compatibility: Helps websites deliver content compatible with the user’s browser capabilities and features.
3. Analytics: Websites and analytics tools use UA strings to track the popularity of different browsers and operating systems among their visitors.

4. Security: UA strings can be used for security purposes, identifying potentially malicious requests or ensuring compatibility with security protocols.

16. What is HyperTest?

HyperTest typically refers to a family of statistical tests used in hypothesis testing, especially in the context of multiple comparisons or simultaneous inference.

In general, when conducting multiple hypothesis tests simultaneously (for example, when comparing several groups or testing several variables), there is an increased risk of making a Type I error (false positive). The term "HyperTest" encompasses various methods and adjustments designed to control this overall error rate.

HyperTest methods are crucial in ensuring that statistical results are reliable and robust when multiple comparisons are involved, preventing the inflation of Type I error rates that would occur if each comparison were tested independently without correction.

17. What is HTML Tags?

HTML tags are the building blocks of an HTML document. They are used to define the structure and content of web pages.

HTML tags:

- Opening and Closing Tags: Most HTML tags come in pairs: an opening tag and a closing tag. The opening tag denotes the beginning of an element, and the closing tag denotes the end. Example: This is a paragraph.
- Attributes: HTML tags can also have attributes, which provide additional information about an element. Attributes are placed within the opening tag and are usually in name-value pairs.
  Example: Visit Example
- Void Elements: Some HTML tags do not have a closing tag and are called void or self-closing elements. For example: <img src="image.jpg" alt="Description">
- Nesting: HTML elements can be nested inside one another. This means you can place one element inside another element. For example: <h1>Welcome to my website!</h1> Here's some content.
- HTML Structure: An HTML document usually begins with a <!DOCTYPE html> declaration followed by an <html> tag that contains a <head> and a <body> section. Inside the <head>, you typically find metadata and references to external resources like stylesheets and scripts. The <body> contains the main content of the document.

18. What is HTML Attributes?

HTML attributes are key-value pairs that provide additional information about HTML elements. They are specified within the opening tag of an element and serve various purposes such as defining element IDs (id), applying styles (class, style), specifying image sources (src), defining input types (type), and enhancing accessibility (title, aria-*). Attributes can be global, applying to most elements, or specific to certain types of elements. They play a crucial role in

structuring and enhancing the functionality and appearance of web pages. Custom attributes prefixed with data- are also commonly used to store extra information that can be accessed programmatically.

19. What is HTML Elements?

HTML elements are the basic building blocks of web pages. An HTML element is defined by a starting tag, some content, and an ending tag (if applicable). Here's a breakdown of the components of an HTML element:

1. Start Tag: This marks the beginning of an element and is enclosed in angle brackets (< and >). It usually contains the name of the element, such as for a paragraph or for a division.
2. End Tag: This marks the end of an element and is also enclosed in angle brackets, but it includes a forward slash (/) before the element name. For example, ends a paragraph element.
3. Content: This is the actual content of the element, which can include text, other elements, images, multimedia, etc. For instance, Hello, world! contains the text "Hello, world!" within a paragraph element.
4. Attributes: These provide additional information about an element and are included within the start tag. Attributes are written as name-value pairs, such as class="container" or src="image.jpg". They modify the behavior or appearance of the element.

20. How do convert elements to tree?

To convert elements into a tree structure, you typically:

1. Define a Node Structure**:** Create a data structure to represent each node in the tree. Each node should contain the element itself and references (pointers) to its child nodes.
2. Build the Tree**:** Starting from the root node, traverse through the elements and construct the tree by assigning child nodes accordingly.
3. Link Nodes**:** Ensure each node is properly linked to its parent and children based on the hierarchical relationship defined by the elements.
4. Handle Edge Cases**:** Consider scenarios like empty elements, duplicate elements, or specific tree structures (e.g., binary tree, n-ary tree) depending on your requirements.
5. Validate and Test**:** Validate the tree structure to ensure it conforms to your expectations and test with different inputs to verify correctness.

This process involves careful handling of data structures and relationships to accurately represent the elements as a tree.

21. What is DOCTYPE?

DOCTYPE stands for Document Type Declaration. It is an instruction or declaration in HTML and XML documents that specifies the type and version of the document. It helps web browsers and other parsers determine how to correctly render or process the document's content. The DOCTYPE declaration is typically placed at the beginning of an HTML document before any other content. Its primary purpose is to ensure proper rendering and parsing of the document by defining its structure and rules.

22. What are the ways we can save html files?

HTML files can be saved in several ways:

1. Text Editor: Use any text editor (e.g., Notepad, Sublime Text, Visual Studio Code) to create and save HTML files with a .html extension.
2. Integrated Development Environment (IDE): IDEs like Visual Studio, IntelliJ IDEA, or Eclipse provide robust features for creating and saving HTML files along with project management capabilities.
3. Content Management Systems (CMS): Platforms like WordPress or Joomla allow you to create and save HTML files through their administration interfaces.
4. Online HTML Editors: Websites such as CodePen, JSFiddle, or JS Bin provide online editors where you can write and save HTML files directly.
5. Browser Developer Tools: You can copy the HTML code from a web page using the "Inspect" tool in browsers and save it as an HTML file on your computer.
6. Command Line: Use command-line tools like echo (Unix/Linux) or type (Windows) combined with output redirection (> or >>) to save HTML content generated programmatically.

23. What is charset? Why we need to use this?

A charset (character set) is a standardized collection of characters used in computing to represent textual data. It defines how characters are encoded as sequences of bits for storage or transmission in digital form. We need to use charsets to ensure that computers and software applications can correctly interpret and display text in various languages and scripts, ensuring compatibility and interoperability across different systems and platforms.

24. What is Metadata? What is the purpose of it?

Metadata refers to data that provides information about other data. In simpler terms, it's data about data. This additional information describes

various attributes of the primary data, such as its content, format, location, authorship, creation date, and so on. The purpose of metadata is to facilitate the understanding, use, management, and discovery of the primary data it describes.

Here are some key purposes and uses of metadata:

1. Identification: Metadata helps uniquely identify and distinguish different pieces of data. It provides labels or tags that make it easier to locate and refer to specific data.
2. Description: Metadata describes the characteristics of data, such as its structure, format, quality, and context. This helps users understand what the data is and how it can be used.
3. Discovery: Metadata aids in discovering and accessing data. It can include keywords, summaries, or abstracts that help users find relevant data among large collections or databases.
4. Management: Metadata supports the management of data throughout its lifecycle. It includes information about ownership, rights, access controls, versioning, and retention policies.
5. Interoperability: Metadata promotes interoperability by providing standardized formats and vocabularies. This allows different systems and applications to exchange and interpret data consistently.
6. Preservation: Metadata helps in preserving data over time by documenting its provenance, history, and relationships with other data.
7. Security: Metadata can include security-related information such as access permissions, encryption details, and audit logs, helping to secure sensitive data.

25. Explain Web Application Architecture?

Web Application Architecture refers to the structure and layout of a web application, detailing how various components interact with each other to deliver functionality to end-users. It encompasses both the front-end (clientside) and back-end (server-side) components, along with the protocols and technologies used for communication between them.

It includes:

1. Client-Side: UI components (HTML, CSS, JavaScript) running in a web browser.
2. Server-Side: Web server, application server, database handling serverside logic.
3. Communication: Protocols like HTTP/HTTPS, WebSockets for clientserver interaction.
4. Integration: APIs (REST, GraphQL) for data exchange.
5. Security: Authentication, authorization, data encryption.
6. Scalability: Load balancing, caching, scaling strategies.
7. Deployment: Cloud services, containers, orchestration for hosting.