Fundamentals Of web Technologies : By Gagan Baghel

Internet

1. The Internet is a global network of computers and devices that are connected to each other, allowing people to share information, communicate, and access websites and services from anywhere in the world.

2. Global network of interconnected computers using some protocols like TCP/IP, Http and https.

3. History :

1969: Birth of ARPANET, the precursor to the Internet. 1983: TCP/IP established as the universal protocol.

1990: Tim Berners-Lee invents the World Wide Web (www), making the Internet accessible to the public.

Intranet

1. An intranet is a private network used within an organization, such as a company or school, to share information, resources, and communication tools among employees or members.

2. Unlike the Internet, an intranet is closed to the public and only accessible to authorized users, often requiring a login.

3. It helps in facilitating internal collaboration, document sharing, and access to company-specific applications.

Extranet

1. An extranet is a private network that allows authorized external users (such as business partners, suppliers, or customers) to access specific parts of an organization's intranet.

2. It enables secure communication and data sharing between a company and its external stakeholders while maintaining control over who can access certain resources.

3. Extranets are commonly used for collaboration, supply chain management, and project coordination.

WWW

1. The World Wide Web (WWW) is a system of interlinked web pages and multimedia content that can be accessed through the Internet.

2. It uses web browsers (like Chrome or Firefox) to display information, and relies on HTTP/HTTPS protocols to transfer data.

3. The WWW was created by Tim Berners-Lee in 1990 and is what most people interact with when they use the Internet to browse websites, watch videos, or read articles.

Web Browser

1. A web browser is a software application that allows users to access, view, and navigate webpages and websites on the Internet.

2. It interprets web pages written in HTML and displays them in a user-friendly format.

3. Popular web browsers include Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

4. Browsers allow you to search for information, stream videos, use web applications, and more.

SOFTWARE

1. Software is a set of instructions or programs that tells a computer how to perform specific tasks.

2. It can range from operating systems (like Windows, macOS) to tools like word processors, games, or development environments.

3. ⇒ Types of Software → 2 Types

1. System Software:
    a. Manages hardware and provides a platform for other software to run (e.g., operating systems like Windows, Linux).
2. Application Software:
    a. Helps users perform specific tasks (e.g., Microsoft Word, Photoshop).

application

1. An application, or "app," is a type of software designed to help users carry out specific tasks, such as writing, browsing, or gaming.

2. Examples of applications include word processors, web browsers, mobile apps, or email clients.

3. Applications are often user-facing, meaning they provide a user interface (UI) for interaction.

summary

⇒ Software is the broader category, encompassing all types of programs.

⇒ Applications are a specific type of software focused on user tasks.

⇒ Software application is another way to describe applications that perform specific functions for users.

⇒ All applications are types of software, but the term "software application" emphasizes its functional use for the user.

search Engine

1. A search engine is an online tool that allows users to search for information on the Internet by entering keywords or phrases.

2. It indexes and retrieves web pages based on relevance to the search query, helping users find the content they are looking for quickly and efficiently.

3. Examples are
    i. Google

ii. Bing

iii. Yahoo

Web page

1. A web page is a document on the World Wide Web that is accessible through a web browser.

2. It is typically written in HTML (Hypertext Markup Language) and can include text, images, videos, links, and other multimedia content

3. ⇒ Types of Web Pages:

   1. Static Web Pages: Display fixed content that does not change unless manually updated by the creator.

   2. Dynamic Web Pages: Generate content dynamically based on user interactions or database queries, often using server-side scripting.

Website

1. ⇒ A website is a collection of related web pages that are hosted on a web server and accessible through the Internet.

2. ⇒ Each website typically has a unique domain name (URL) that users can enter into a web browser to access it.

3. ⇒ Types → static & Dynamic

URL – Uniform Resource Locator

1. Definition

A URL, or Uniform Resource Locator, is a specific address used to access resources on the Internet. It provides a way to identify and locate resources such as web pages, images, videos, and files.

2. Structure of a URL

A URL follows the structure:

scheme://username:password@hostname:port/path?query#fragment

1. **Scheme**: Indicates the protocol used to access the resource (e.g., HTTP, HTTPS, FTP).
    - Example: `https://` (Hypertext Transfer Protocol Secure)
2. **Example of a complete URL**:
    - `https://www.example.com:443/folder/page.html?search=query#section1`
3. **Username (optional)**: Credentials for authentication to access a resource.
    - Example: `user:pass@` (rarely used in modern URLs)
4. **Hostname**: The domain name or IP address of the server hosting the resource.
    - Example: `www.example.com` or `192.0.2.1`
5. **Port (optional)**: The port number on the server to connect to. Default ports are often omitted (e.g., port 80 for HTTP, port 443 for HTTPS).
    - Example: `:8080`
6. **Path**: The specific location of the resource on the server, which may include directories and filenames.
    - Example: `/folder/page.html`
7. **Query (optional)**: A string of parameters for the request, typically used for dynamic content, starting with a question mark (`?`).
    - Example: `?search=keyword&sort=asc`
    - **Query Parameters**: Content after `=` (e.g., `keyword`).
    - **Query Equal Operator**: `=`
8. **Fragment (optional)**: A specific section of a web page, indicated by a hash symbol (`#`).
    - Example: `#section1`

3. Types of URLs

1. **Absolute URL**: A complete URL that includes the full path to the resource.
    - Example: `https://www.example.com/folder/page.html`

2. **Relative URL**: A partial URL that does not include the domain name, often used within the same website.
    - Example: `/folder/page.html` (relative to the current domain)

4. Importance of URLs

1. Navigation
2. Linking
3. SEO
    - Keep it simple
    - Use hyphens
    - Avoid special characters

---

11. Protocols and Their Types

1. Definition

- Protocols are a set of rules and conventions that govern how data is transmitted and received over a network.
- They ensure effective communication between devices and applications.

2. Purpose

- To facilitate interoperability between different hardware and software systems.
- To ensure reliable and secure data transfer.

3. Types of Protocols

- **Communication Protocols**: Define how data is formatted and transmitted (e.g., TCP/IP, HTTP, FTP).
- **Network Protocols**: Manage how devices on a network communicate (e.g., IP, ICMP).
- **Application Protocols**: Specify how applications communicate over the network (e.g., SMTP for email, DNS for domain resolution).

4. Key Protocols

- **HTTP (Hypertext Transfer Protocol)**:
  - Used for transmitting web pages over the Internet.
  - The foundation of data communication for the World Wide Web.
- **HTTPS (HTTP Secure)**: A secure version of HTTP that encrypts data exchanged between the browser and server.
- **FTP (File Transfer Protocol)**:
  - Used for transferring files between a client and server over the Internet.
  - Allows for file uploads and downloads, with options for anonymous access.
- **TCP (Transmission Control Protocol)**:
  - Works on top of IP to ensure reliable, ordered, and error-checked delivery of data between applications.
  - Establishes a connection before transmitting data and ensures that packets are received in the correct order.
- **IP (Internet Protocol)**: Responsible for addressing and routing packets of data across networks.

---

12. IP – Internet Protocol

- Responsible for addressing and routing packets of data across networks.
- Has two main versions:
  - **IPv4**: Uses a 32-bit address scheme, allowing for approximately 4.3 billion unique addresses.
  - **IPv6**: Uses a 128-bit address scheme, allowing for a vastly larger number of addresses to accommodate the growing number of devices.

---

13. DNS – Domain Name System

1. Definition

The Domain Name System (DNS) is a hierarchical and decentralized naming system used to translate human-readable domain names (like `www.example.com`) into machine-readable IP addresses (like `192.0.2.1`).

2. Purpose

- To make it easier for users to access websites and services without remembering complex numerical IP addresses.
- To provide a structured way of organizing and managing domain names.

3. How DNS Works

1. When a user types a domain name into a web browser, a DNS query is sent to a DNS resolver (usually provided by the user's Internet Service Provider).
2. The resolver checks its cache for the corresponding IP address. If not found, it queries other DNS servers hierarchically:
    - **Root DNS Servers**: The resolver first queries a root DNS server, which directs it to a top-level domain (TLD) server (e.g., `.com`, `.org`).
    - **TLD DNS Servers**: The resolver then queries the TLD server, which points to the authoritative DNS server for the specific domain.
    - **Authoritative DNS Server**: The authoritative server holds the DNS records for the domain and provides the corresponding IP address.
3. **Caching**: The resolver caches the IP address for a specified time (Time to Live or TTL) to speed up future requests for the same domain.

4. Types of DNS Records

- **A Record**: Maps a domain name to its corresponding IPv4 address.
- **AAAA Record**: Maps a domain name to its corresponding IPv6 address.
- **CNAME Record**: Alias for another domain name, allowing multiple domain names to point to a single IP address.
- **MX Record**: Specifies the mail exchange servers for handling email for a domain.

- **NS Record**: Indicates the authoritative name servers for the domain.

5. DNS Hierarchy

- **Root Level**: The top of the DNS hierarchy, represented by a dot (`.`) and managed by root DNS servers.
- **TLD Level**: Contains top-level domains such as `.com`, `.org`, `.net`, etc.
- **Second-Level Domains**: The part of the domain name that comes before the TLD (e.g., `example` in `example.com`).
- **Subdomains**: Additional divisions within a domain (e.g., `www` in `www.example.com`).

6. Importance of DNS

- **User-Friendly**: Translates complex IP addresses into easy-to-remember domain names.
- **Load Distribution**: Supports multiple IP addresses for a single domain, allowing for load balancing and redundancy.
- **Email Routing**: Directs email traffic to the correct mail servers through MX records.

Database

1. **Definition**:
   - A database is an organized collection of structured information or data, typically stored electronically in a computer system. It allows for efficient data management, retrieval, and manipulation.
2. **Purpose**:
   - To store, retrieve, and manage data effectively.
   - To support various applications by providing a structured way to access and manipulate data.
3. **Types of Databases**:

- **Relational Databases**:
  - Organize data into tables (relations) that can be linked through common fields.
  - Use Structured Query Language (SQL) for data manipulation and querying.
  - **Examples**: MySQL, PostgreSQL, Oracle Database.
- **NoSQL Databases**:
  - Designed for unstructured or semi-structured data, often allowing for horizontal scaling.
  - Types include document databases, key-value stores, column-family stores, and graph databases.
  - **Examples**: MongoDB (document), Redis (key-value), Cassandra (column-family).
- **In-Memory Databases**:
  - Store data in RAM for faster access, often used for caching and real-time applications.
  - **Examples**: Redis, Memcached.
- **NewSQL Databases**:
  - Combine the scalability of NoSQL with the ACID guarantees of traditional SQL databases.
  - **Examples**: Google Spanner, CockroachDB.

4. **Key Components**:
   - **Data**: The actual information stored in the database.
   - **Database Management System (DBMS)**: Software that interacts with the database, allowing users to create, read, update, and delete data.
     - Examples of DBMS: MySQL, Oracle, MongoDB, Microsoft SQL Server.
   - **Schema**: The structure that defines how data is organized in the database, including tables, fields, and relationships.

5. **Database Operations**:
    - **CRUD Operations**: Basic functions for managing data: Create, Read, Update, and Delete.
    - **Querying**: Using a query language (like SQL) to retrieve specific data from the database.
    - **Transactions**: A sequence of operations performed as a single unit, ensuring data integrity through ACID properties (Atomicity, Consistency, Isolation, Durability).
6. **Importance of Databases**:
    - Enable efficient data storage, retrieval, and manipulation for various applications.
    - Support data integrity, security, and consistency across applications and users.
    - Allow for data analysis and reporting, helping organizations make informed decisions.
7. **Conclusion**:
    - A database is a fundamental component of modern software applications, providing a structured way to store and manage data. Its organization, efficiency, and support for complex data relationships make it essential for various industries and applications.

Server

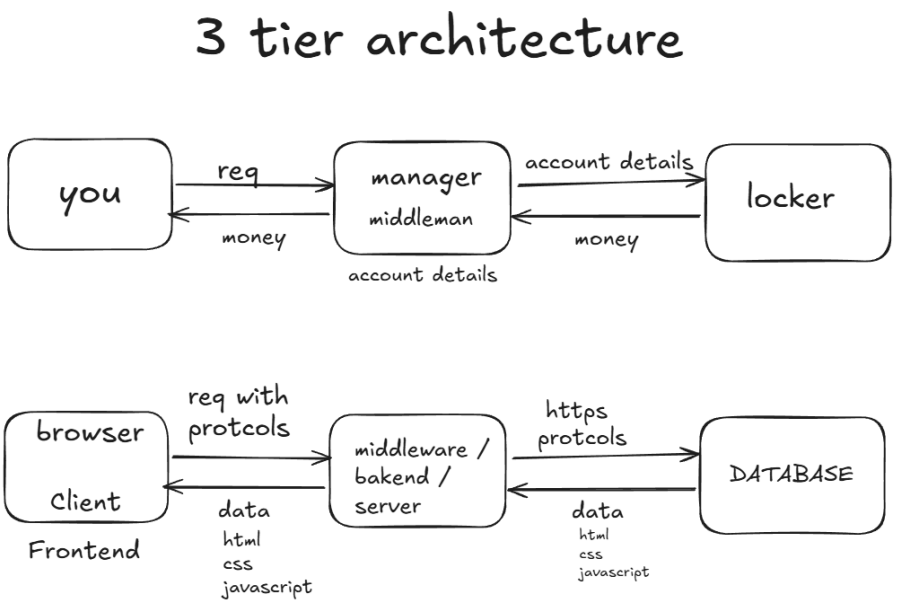
1. **Definition**:
    - A server is a specialized computer or system that provides resources, data, services, or programs to other computers, known as clients, over a network.
2. **Purpose**:
    - To manage network resources and provide various services to client devices, including file storage, database management, web hosting, and application execution.

3. **Types of Servers**:
    - **Web Server**:
        - Hosts websites and serves web pages to clients via the Internet using protocols like HTTP or HTTPS.
        - **Examples**: Apache, Nginx, Microsoft Internet Information Services (IIS).
    - **Database Server**:
        - Provides database services to other computers over a network, allowing for data storage, retrieval, and management.
        - **Examples**: MySQL Server, Oracle Database Server, Microsoft SQL Server.
    - **File Server**:
        - Stores and manages files, allowing clients to access, share, and manage data across a network.
    - **Application Server**:
        - Hosts and runs applications, providing business logic and services to client applications.
        - Often used in web application architectures to process user requests.
    - **Mail Server**:
        - Manages email communication by sending, receiving, and storing email messages.
        - **Examples**: Microsoft Exchange Server, Postfix.
    - **Proxy Server**:
        - Acts as an intermediary between clients and other servers, providing anonymity, security, and caching of web content.
4. **Server Architecture**:
    - **Hardware**: Physical components (CPU, RAM, storage) designed to handle multiple requests and high workloads.

- **Operating System**: Server operating systems (e.g., Linux, Windows Server) that manage hardware resources and provide server functionality.
- **Network Interface**: Facilitates communication between the server and clients over a network.

5. **Key Characteristics**:
   - **High Availability**: Servers are often designed for continuous operation, minimizing downtime.
   - **Scalability**: Capable of handling increased loads by adding resources or upgrading hardware.
   - **Reliability**: Built to operate consistently under high demand and provide fault tolerance.
6. **Importance of Servers**:
   - Centralize data and application management, making it easier to maintain and secure resources.
   - Enable collaboration and communication among users and applications across networks.
   - Support cloud computing and virtualization, allowing for resource pooling and efficient usage.
7. **Conclusion**:
   - Servers play a crucial role in the functioning of networks and the Internet, providing essential services and resources to clients. Their ability to manage data, applications, and communications makes them a foundational element in modern computing environments.

▼ 3 Tier Architecture



How the Web Works - In-Depth

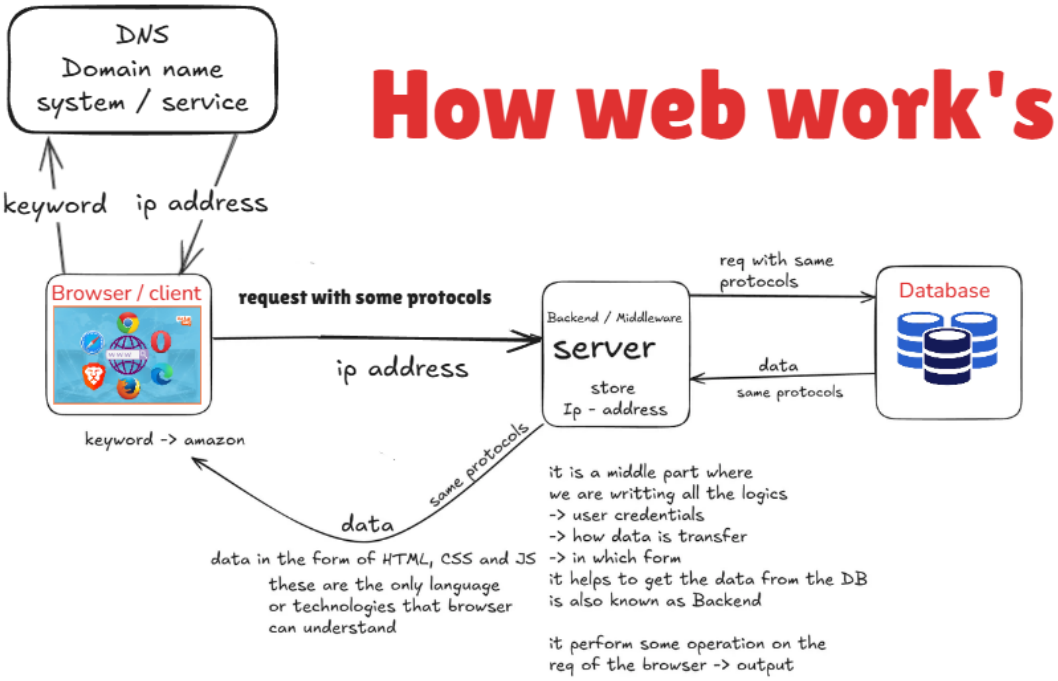
1. **Overview of the Web**

- The web, or World Wide Web (WWW), is a system of interlinked hypertext documents accessed via the Internet.
- Users can view and interact with web pages containing text, images, videos, and other multimedia through web browsers.

2. **Key Components**

- **Web Browsers**: Applications that retrieve, display, and interact with content on the web (e.g., Chrome, Firefox, Safari).
- **Web Servers**: Computers that store and serve web pages to users upon request.
- **Web Protocols**: Rules for communication over the web, primarily HTTP (Hypertext Transfer Protocol) and HTTPS (HTTP Secure).

3. **How the Web Works: Step-by-Step Process**



A. URL Resolution

1. **User Input**:
   - A user enters a URL (Uniform Resource Locator) in the browser's address bar (e.g., https://www.example.com).
2. **DNS Lookup**:
   - The browser initiates a DNS (Domain Name System) query to translate the domain name into an IP address.
   - The DNS resolver queries root servers, TLD servers, and authoritative servers to obtain the IP address of the target web server.

B. Establishing a Connection

1. **TCP Connection**:
   - The browser establishes a TCP (Transmission Control Protocol) connection with the web server using the obtained IP address.
   - This involves a three-way handshake process (SYN, SYN-ACK, ACK) to ensure reliable communication.

2. **TLS/SSL Handshake (for HTTPS)**:
    - If the connection is secure (HTTPS), a TLS (Transport Layer Security) or SSL (Secure Sockets Layer) handshake occurs to establish an encrypted connection.

C. Sending an HTTP Request

1. **HTTP Request**:
    - The browser sends an HTTP request to the server, specifying the desired resource (e.g., a web page).
    - The request includes:
        - Request method (e.g., GET, POST)
        - Headers (metadata about the request)
        - (Optional) Body (for methods like POST)

D. Server Processing

1. **Server Processing**:
    - The web server receives the HTTP request and processes it.
    - If the requested resource is found, the server prepares an HTTP response.
    - If not found, the server sends an error response (e.g., 404 Not Found).

E. Sending an HTTP Response

1. **HTTP Response**:
    - The server sends an HTTP response back to the browser, which includes:
        - Status code (indicating success or failure)
        - Headers (metadata about the response)
        - Body (the requested content, typically HTML, CSS, JavaScript)

F. Rendering the Web Page

1. **Rendering**:

- The browser receives the HTTP response and begins rendering the web page.
- The process involves:
  - Parsing HTML: The browser constructs the Document Object Model (DOM).
  - Applying CSS: The browser applies styles to the elements.
  - Executing JavaScript: The browser runs any JavaScript code, which may modify the DOM or make additional requests.

2. **Loading Additional Resources**:
   - The initial page load may trigger additional requests for resources (e.g., images, CSS files, JavaScript files).
   - These resources are fetched similarly through DNS resolution and HTTP requests.

4. Web Technologies Involved

- **HTML (Hypertext Markup Language)**: The standard markup language for creating web pages and applications.
- **CSS (Cascading Style Sheets)**: Styles the appearance of web pages (layout, colors, fonts).
- **JavaScript**: A programming language that adds interactivity and dynamic content to web pages.
- **Web Frameworks**: Tools and libraries (e.g., React, Angular, Vue.js) that facilitate web application development.

5. Client-Server Model

- The web operates on a client-server architecture:
  - **Client**: The user's device running a web browser that initiates requests for resources.
  - **Server**: The remote machine that hosts

the resources and responds to requests.

6. **Conclusion**

- The web relies on a complex interplay of protocols, technologies, and services to deliver information to users. Understanding how the web works allows developers to optimize performance, enhance security, and deliver better user experiences.

---

17. Web application or website Understanding → Concept of HTML, CSS & JS

The websites are widely made by using the core technologies like html, CSS and JS

1. HTML: it stands for Hypertext markup Language, it is used to create the structure of the website
2. CSS: Cascading Style sheet, it is used to style or beautify the website, therefore it helps to increase the user engagement and make website more visually appealing
3. JS: JavaScript is helps to add the functionalities in our website, therefore it makes for website dynamic as well as add the interaction that can be done by user whenever they visit in our website

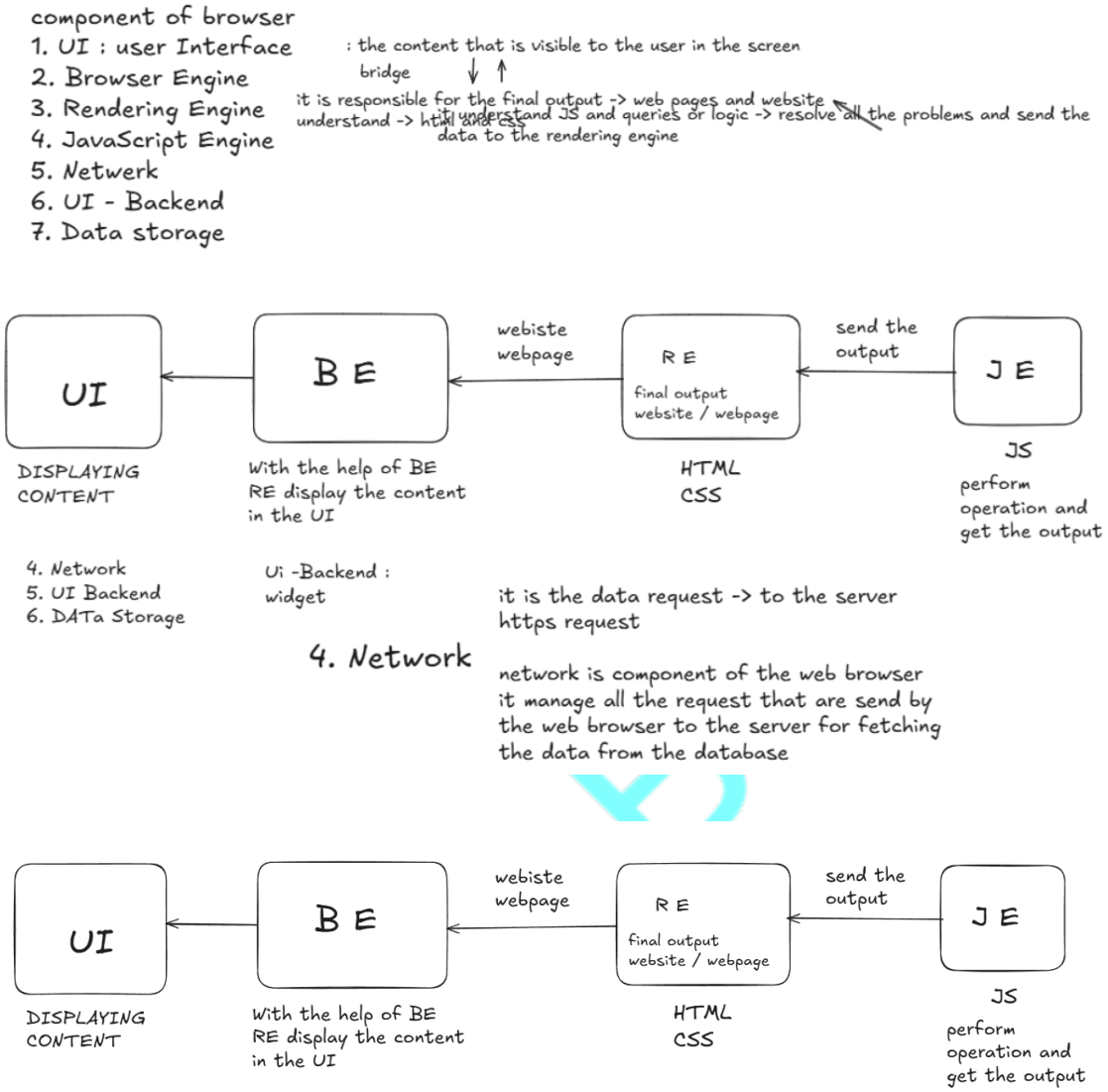
html - understand by rendering engine that is the component of the browser

CSS - understand by rendering engine that is the component of the browser

JS - understand by JavaScript interpreter that is the component of the browser

React JS - Uses JSX

Key Components of a Web Browser (6 Main Components)



1. **User Interface (UI)**:
   - The visual layout that allows users to interact with the browser, including the address bar, navigation buttons, and tabs.
2. **Rendering Engine**:
   - Responsible for displaying web pages.
   - It interprets HTML and CSS to create the layout and render the content for viewing.

- responsible for rendering a specific web page requested by the user on their screen.
- It interprets html and xml documents along with images that are styled or formatted using CSS, and a final layout is generated, which is displayed on the user interface
- Different browsers have different rendering engine like
    1. Google chrome & opera -Blink
    2. Internet Explorer - Trident
    3. Mozilla fire Fox - Gecko
    4. Chrome for IOS & safari - Web kit

3. **JavaScript Engine**:
    - Executes JavaScript code on web pages, enabling dynamic content and interactivity. It converts JavaScript into executable code.
    - it is responsible for parsing and executing the JavaScript code embedded in a website once the interpreted results are generated. they are forwarded to the rendering engine for display on the user interface
4. **Networking Component**:
    - Manages all network requests and protocols (like HTTP/HTTPS) to fetch resources from the web, including DNS resolution.
5. **Cache**:
    - Stores copies of previously accessed web pages and resources locally to improve loading speed and reduce bandwidth usage for frequently visited sites.
6. **Storage**:
    - Manages local storage options for web applications, such as cookies, session storage, and local storage, allowing web apps to save data on the client-side.

Conclusion

These six components work together to provide a seamless browsing experience, allowing users to access and interact with web content efficiently. Understanding these components helps in grasping how browsers function and interact with web technologies.