1.WWW and Web development

1. What is the world wide web (www) and how does it differ from the internet?

The World Wide Web -- also known as the web, WWW - refers to all the public websites or pages that users can access on their local computers and other devices through the internet. These pages and documents are interconnected by means of hyperlinks that users click on for information.

The Internet and the World Wide Web (WWW) are often used interchangeably, but they are not the same. The Internet is a global network of interconnected computer networks, while the WWW is a specific service that operates on top of the Internet, allowing users to access information through websites and web browsers. The Internet provides the infrastructure, while the WWW is the application that uses it.

The Internet:

* **Definition:** A vast global network of interconnected computers and devices.
* **Purpose:** Facilitates communication and data transfer between these interconnected systems.
* **Infrastructure:** The underlying hardware and software that enable communication.
* **Examples:** Email, file transfer, and various other applications that use the network.

The World Wide Web (WWW):

* **Definition:** An information system that provides access to documents and other resources via the Internet, using the Hypertext Transfer Protocol (HTTP).
* **Purpose:** Enables users to browse and interact with websites and web pages.
* **Application:** A service that operates on top of the Internet, using the Internet's infrastructure to transmit data.
* **Examples:** Web pages, websites, and the information they contain.

1. Explain the basic structure of a web page. What are the essential elements?

A webpage's structure is fundamentally organized around HTML elements, with the basic components being a <!DOCTYPE html> declaration, an <html> element, a <head> section for metadata, and a <body> section for visible content. Within the body, common sections include a header, navigation bar, main content, sidebar, and footer.

* **Header:** Typically placed at the top of the page and often includes the website's logo and main title.
* **Navigation Bar:** Provides navigation links to different sections or pages within the site.
* **Main Content:** The core content of the webpage, which can be further divided into sections or articles.
* **Sidebar:** Often used to display supplementary information, navigation menus, or ads.
* **Footer:** Located at the bottom of the page and usually contains copyright information, privacy policies, and contact details.

Essential elements within this structure include the head (containing metadata and title) and the body (containing the visible content).

1. <!DOCTYPE html>:

This declaration specifies that the document is an HTML5 document, indicating the version of HTML being used.

2. <html>:

The root element of the HTML page. Everything inside this element is considered part of the HTML document.

3. <head>:

Contains meta information about the HTML page that isn't displayed on the page itself, such as the page title, character set, links to CSS files, and other instructions for the browser.

* + <title>: Specifies a title for the HTML page, which is typically displayed in the browser's title bar or tab.

4. <body>:

Defines the visible content of the HTML page, including text, images, links, and other elements.

5. Other Essential Elements:

* + <h1> - <h6>: Heading elements for structuring text on the page, with <h1> being the largest and <h6> the smallest.
  + <p>: Paragraph element for formatting blocks of text.
  + <img>: Image element for including images on the page.
  + <a>: Anchor element for creating hyperlinks.
  + <ul>, <ol>, <li>: List elements for creating unordered and ordered lists.
  + <table>, <tr>, <td>: Table elements for structuring data in a table format.

c. What is the role of HTML, CSS, and JavaScript in web development?

In web development, HTML defines the content, CSS controls the layout and presentation, and JavaScript handles the dynamic behavior of web pages. HTML provides the structure, CSS styles it, and JavaScript adds interactivity.

* **HTML (HyperText Markup Language):**

This is the foundation of web pages, defining the structure and content. It uses tags to organize elements like headings, paragraphs, images, links, and more.

* **CSS (Cascading Style Sheets):**

CSS is used to style the HTML elements, controlling aspects like colors, fonts, spacing, and layout. It separates the presentation of a web page from its content, making it easier to manage and update styles.

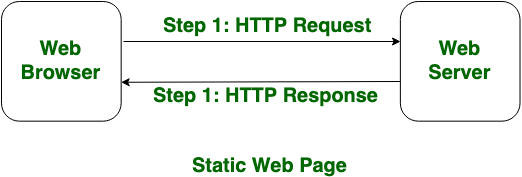
* **JavaScript:**

This programming language adds interactivity and dynamic behavior to web pages. It can manipulate HTML elements, update content, respond to user actions, and much more.

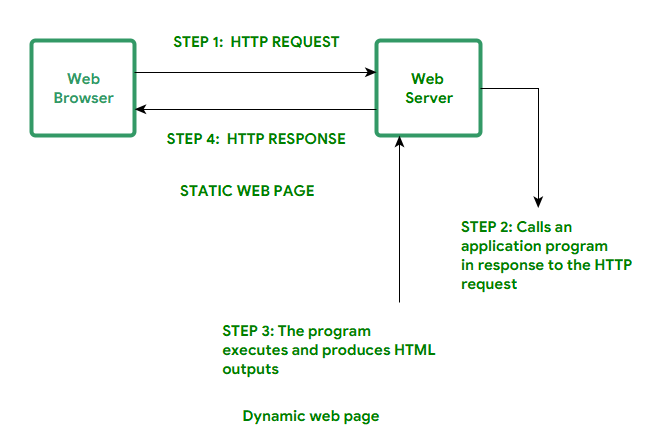
d. Describe the difference between static and dynamic web pages.

There are two basic methods of web design: static and dynamic web pages. Users access static web pages, which present the same content every time they are viewed. On the other hand, dynamic webpages create content instantly in response to user input and present customized or updated information. Let’s learn about this in detail in this article.

**What are Static Web Pages?**

Static Web pages are very simple. It is written in languages such as HTML, JavaScript, CSS, etc. For static web pages when a server receives a request for a web page, then the server sends the response to the client without doing any additional process. These web pages are seen through a web browser. In [static web pages](https://www.geeksforgeeks.org/static-websites/), Pages will remain the same until someone changes it manually.   


**What are Dynamic Web Pages?**

Dynamic Web Pages are written in languages such as CGI, AJAX, ASP, ASP.NET, etc. In dynamic web pages, the Content of pages is different for different visitors. It takes more time to load than the static web page. [Dynamic web pages](https://www.geeksforgeeks.org/dynamic-websites/) are used where the information is changed frequently, for example, stock prices, weather information, etc.   


**Difference Between Static and Dynamic Web Pages**

| **Static Web Page** | **Dynamic Web Page** |
| --- | --- |
| In static web pages, Pages will remain same until someone changes it manually. | In dynamic web pages, Content of pages are different for different visitors. |
| Static Web Pages are simple in terms of complexity. | Dynamic web pages are complicated. |
| In static web pages, Information are change rarely. | In dynamic web page, Information are change frequently. |
| Static Web Page takes less time for loading than dynamic web page. | Dynamic web page takes more time for loading. |
| In Static Web Pages, database is not used. | In dynamic web pages, database is used. |
| Static web pages are written in languages such as: HTML, JavaScript, CSS, etc. | Dynamic web pages are written in languages such as: CGI, AJAX, ASP, ASP.NET, etc. |
| Static web pages does not contain any application program . | Dynamic web pages contains application program for different services. |
| Static web pages require less work and cost in designing them. | Dynamic web pages require comparatively more work and cost in designing them. |

2. Web Applications and Types of Web Applications

a. What is a web application, and how does it differ from a website?

A web application is a software program accessed through a web browser over a network, like the internet. Unlike traditional apps that require installation, web applications are hosted on a remote server and can be accessed directly from any device with a web browser, eliminating the need for separate software installations.

* Examples of web applications include online banking, e-commerce platforms, social media networks, and online games.

Below is a table of differences between Web Application and Website:

| **Web Application** | **Website** |
| --- | --- |
| Web application is designed for interaction with end users. | Website basically contains static content. |
| The user of web application can read the content of web application and also manipulate the data. | The user of website only can read the content of website but not manipulate . |
| The web application site should be precompiled before deployment. | The website does not need to be precompiled . |
| The function of a web application is quite complex. | The function of website is simple. |
| Web application is interactive for users. | Web site is not interactive for users. |
| The browser capabilities involved with a web application is high. | The browser capabilities involved with web site is high. |
| Integration is complex for web application because of its complex functionality. | Integration is simpler for web site. |
| Web application mostly requires authentication | In web site authentication is not necessary. |
| EXAMPLE :- Amazon, Facebook, etc. | EXAMPLE :- Breaking News, Aktu website, etc. |

b.Name and describe three different types of web applications.

 Three main types of web applications are: static web applications, dynamic web applications, and single-page applications (SPAs).

1. **1. Static Web Applications:**

These applications deliver pre-defined content to users without any dynamic changes based on user interaction or server-side processing. They are typically built with HTML, CSS, and JavaScript and are suitable for simple, information-heavy websites that don't require user input or database interaction.

1. **2. Dynamic Web Applications:**

These applications generate content on the fly based on user interaction or server-side logic. They often involve database interactions and can be tailored to individual user needs. Examples include webmail, online shopping platforms, and social media sites.

1. **3. Single-Page Applications (SPAs):**

SPAs are a type of dynamic application that loads a single HTML page and dynamically updates the content within that page without requiring full page reloads. They offer a more fluid and app-like user experience and are often built using JavaScript frameworks like React or Angular.

c.What are the advantages and disadvantages of Single Page Applications (SPAs) compared to Multi-Page Applications(MPAs)?

Single Page Applications (SPAs) generally offer faster loading times, better user experience due to smoother transitions, and improved offline functionality, while Multi-Page Applications (MPAs) excel in SEO and are easier to develop initially. SPAs can be more complex to develop and manage, and their SEO can be more challenging compared to MPAs.

Advantages of SPAs:

* **Faster loading times:**

SPAs only load a single HTML page initially, with content updated dynamically using JavaScript, leading to faster page transitions and a more fluid user experience.

* **Enhanced user experience:**

Dynamic content updates without page reloads create a more app-like feel, making navigation smoother and more responsive.

* **Better offline functionality:**

SPAs can cache data locally, allowing users to continue interacting with the application even when offline.

* **Easier to debug:**

Development tools like Chrome DevTools can be used effectively to debug SPAs, making development and troubleshooting easier.

Disadvantages of SPAs:

* **SEO challenges:**

SPAs can be more challenging to optimize for search engines, as crawlers might not be able to easily index dynamically generated content.

* **More complex development:**

SPAs often require a deeper understanding of JavaScript frameworks and AJAX, leading to a steeper learning curve for developers.

* **Potentially slower initial load:**

While SPAs offer fast transitions, the initial load time can be longer as all the necessary code and data are loaded at once.

* **Challenges with browser history and back/forward buttons:**

SPAs can sometimes have issues with accurately managing browser history and back/forward button navigation, which might require careful implementation.

Advantages of MPAs:

* **Better SEO:**

MPAs are inherently easier to index for search engines, as each page has its own HTML structure, making them more SEO-friendly.

* **Easier development:**

MPAs are typically easier to develop and maintain compared to SPAs, as they don't require a strong understanding of JavaScript frameworks and AJAX.

* **Faster initial load:**

MPAs can have faster initial load times, as they only load the necessary HTML for each page, rather than a full application code base.

* **Better browser history and back/forward button support:**

MPAs naturally support browser history and back/forward buttons, making navigation easier for users.

Disadvantages of MPAs:

* **Slower page load times:**

MPAs can be slower as they require loading a new page for every navigation, leading to slower transitions and a less fluid user experience.

* **Potentially slower performance:**

Since MPAs load new pages with every navigation, they can experience slower overall performance, especially on slower connections.

* **Limited offline functionality:**

MPAs generally don't offer the same level of offline functionality as SPAs, as they require a server connection to load new pages.

* **Less flexible user experience:**

MPAs can sometimes feel less flexible and app-like compared to SPAs, as they require page reloads for every navigation.

3.Web Client and Web Server

a. What is a web client, and how does it interact with a web server?

A [Web client](https://cloudinfrastructureservices.co.uk/how-to-setup-apache-webdav-server-access-on-ubuntu-20-04/) is an application installed on the user’s device that they can use to surf the internet. Web clients request computer servers for a webpage but don’t store them.

Without these clients, an ordinary user can’t use the internet. When you search for a particular webpage through your browser, it retrieves the page from the appropriate server and displays the result. The client and server communicate via [HTTP](https://cloudinfrastructureservices.co.uk/ubuntu-vs-pop_os-whats-the-difference/)(Hypertext Transfer Protocol).

 HTTP protocol as the language of the web. It enables networked devices to send and receive traffic using a set of simple rules. However, there’re other types of protocols, such as [SSH](https://cloudinfrastructureservices.co.uk/sftp-vs-ssh-vs-ftp-vs-ftps/)(Secure Shell), [FTP](https://cloudinfrastructureservices.co.uk/nfs-vs-ftp-whats-the-difference/)(File Transfer Protocol) and [SMTP](https://cloudinfrastructureservices.co.uk/simple-mail-transfer-protocol-explained-what-is-smtp-and-how-it-works/)(Simple Mail Transfer Protocol).

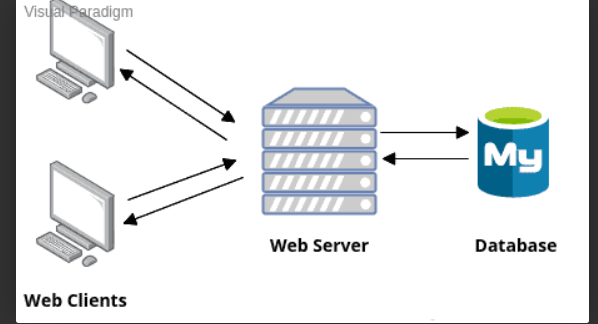
Examples of web clients include Google Chrome, Microsoft Edge, Safari, Tor, etc.

A [Web Servers](https://cloudinfrastructureservices.co.uk/how-to-setup-apache-webdav-server-access-on-ubuntu-20-04/) are systems that are connected to the internet and store web pages. In addition it sends out requested data over the web using HTTP. In nutshell, like [Apache](https://httpd.apache.org/)– the [Web Servers](https://cloudinfrastructureservices.co.uk/what-is-apache-mpm-multi-processing-modules-and-how-they-work/) are just like libraries for web pages. They store, process and deliver the web resources to the client software.

Servers can also use other data transfer protocols such as [SMTP](https://cloudinfrastructureservices.co.uk/how-to-setup-hmailserver-email-server-in-azure-aws-gcp/) and [FTP](https://cloudinfrastructureservices.co.uk/setup-ftp-server-secure-on-azure-windows/). Mail servers use [SMTP](https://cloudinfrastructureservices.co.uk/smtp-vs-imap-whats-the-difference/) for transferring emails from one user to another. Additionally [FTP](https://cloudinfrastructureservices.co.uk/how-to-setup-ftp-server-on-amazon-aws-windows-ec2-instance/) server is used for high speed file transfer operations.

Older web servers used to serve the requested documents as is, without any modifications. These types of servers are known as static servers. Most modern servers today serve dynamic content instead. Dynamic web servers modify the requested content on the fly.

The [Web server](https://cloudinfrastructureservices.co.uk/how-to-setup-apache-web-server-mysql-server-on-linux-in-azure-aws-gcp/) applications include a range of software that provide features like sending and receiving client requests, processing data on the fly and displaying them in the correct order. These applications communicate with your server via the HTTP protocol. However, some servers also use other protocols.

­­

When you send a web request to a page, one of these server applications handles your request. It checks if the request made by the client software is valid and whether the requested information is available on the server or not. Once confirmed, the web server application processes the request and sends the desired information to the web client.

b. Explain the role of a web server in delivering web content to users.

A web server's primary role is to host websites and deliver web content to users over the internet. It acts as a central hub, receiving requests from web browsers, processing them, and sending back the requested web pages, images, and other files. This process ensures users can access and view website content.

c.What is the purpose of a server -side language, and how does it differ from client -side language?

Server-side scripting languages are used to generate dynamic web pages, interact with databases, handle user authentication, and perform other tasks that require server-side processing. Examples of popular server-side scripting languages include PHP, Python, Ruby, Java, and Node. js.

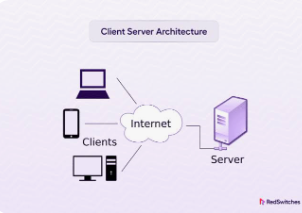
**Difference between client-side scripting and server-side scripting :**

| **Client-side scripting** | **Server-side scripting** |
| --- | --- |
| Source code is visible to the user. | Source code is not visible to the user because its output  of server-sideside is an HTML page. |
| Its main function is to provide the requested output to the end user. | Its primary function is to manipulate and provide access to the respective database as per the request. |
| It usually depends on the browser and its version. | In this any server-side technology can be used and it does not  depend on the client. |
| It runs on the user’s computer. | It runs on the webserver. |
| There are many advantages linked with this like faster.  response times, a more interactive application. | The primary advantage is its ability to highly customize, response  requirements, access rights based on user. |
| It does not provide security for data. | It provides more security for data. |
| It is a technique used in web development in which scripts run on the client’s browser. | It is a technique that uses scripts on the webserver to produce a response that is customized for each client’s request. |
| HTML, CSS, and javascript are used. | PHP, Python, Java, Ruby are used. |
| No need of interaction with the server. | It is all about interacting with the servers. |
| It reduces load on processing unit of the server. | It surge the processing load on the server. |

4.Client-Server Communication

a. Describe the process of client-server communication in a typical web application.

In a typical web application, client-server communication works through a request-response cycle. The client (like a web browser) initiates the communication by sending a request to the server, usually formatted using a protocol like HTTP. The server receives the request, processes it, and then sends back a response, which could be a web page, data, or an acknowledgment.



b. What is a RESTful API, and how does it facilitate client-server communication?

A RESTful API is an architectural style for application programming interfaces (APIs) that uses the principles of Representational State Transfer (REST). It facilitates client-server communication by providing a uniform interface and using standard HTTP methods to interact with resources. This uniformity and standardization make RESTful APIs simpler to use, more scalable, and easier to maintain than some older API styles.

How RESTful APIs facilitate client-server communication:

**1. Uniform Interface:**

REST APIs use a standardized approach where clients and servers communicate using HTTP methods (GET, POST, PUT, DELETE, etc.) and URLs to identify resources. This uniform interface allows clients to interact with resources consistently, regardless of the underlying server implementation.

**2. Client-Server Architecture:**

REST APIs operate under the principle of a client-server architecture, where clients (applications or services) request resources from servers. This separation of concerns allows clients and servers to evolve independently, making the system more flexible and scalable.

**3. Statelessness:**

REST APIs are stateless, meaning that the server does not store any information about the client's state between requests. Each client request contains all the necessary information for the server to process, making it simpler to handle multiple clients and scale the system.

**4. Cacheable:**

REST APIs allow for caching, which can improve performance by storing frequently accessed data locally on the client's side.

**5. Layered System:**

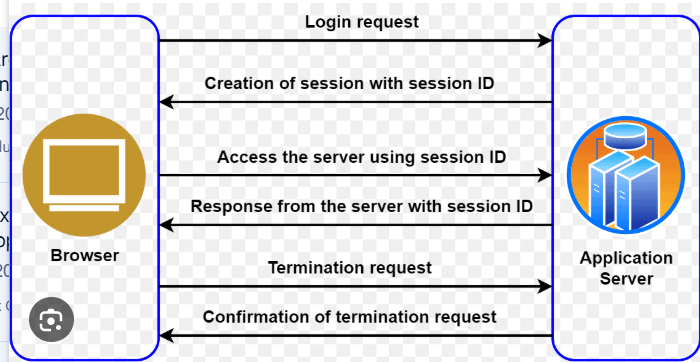
REST APIs can be layered, allowing for the use of multiple servers or layers to perform tasks such as load balancing, security, or content delivery.

**6. Code on Demand:**

REST APIs can support the principle of code on demand, where servers can send executable code to clients to be executed on the client's side.

c.Explain the concept of a session in the context of web development.

In web development, a session represents a series of related requests made by a user to a website within a specific timeframe. It's a way to maintain state across multiple HTTP requests, allowing websites to remember information about a user even though HTTP is inherently stateless. Think of it as a temporary storage for data associated with a particular visitor.



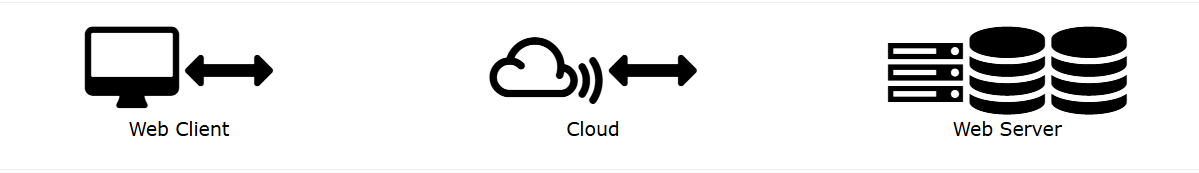
5.HTTP and HTTP Methods

a. What is HTTP, and why is it essential for web communication?

HTTP stands for Hyper Text Transfer Protocol. WWW is about communication between web clients and servers. Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses

World Wide Web Communication

The World Wide Web is about communication between web clients and web servers. Clients are often browsers (Chrome, Edge, Safari), but they can be any type of program or device. Servers are most often computers in the cloud.



HTTP Request / Response

Communication between clients and servers is done by requests and responses:

1. A client (a browser) sends an HTTP request to the web
2. A web server receives the request
3. The server runs an application to process the request
4. The server returns an HTTP response (output) to the browser
5. The client (the browser) receives the response.

HTTP is a method for encoding and transporting information between a client (such as a web browser) and a web server. HTTP is the primary protocol for transmission of information across the internet. Information is exchanged between clients and servers in the form of hypertext documents, from which HTTP gets its name.

b. List and describe the different HTTP methods (e.g., GET, POST, PUT, DELETE). When should each be used?

The most commonly used HTTP methods are:

GET: The GET method is used to retrieve data on a server. ...

POST: The POST method is used to create new resources. ...

PUT: The PUT method is used to replace an existing resource with an updated version. ...

PATCH: The PATCH method is used to update an existing resource.

DELETE: The DELETE method is used to remove data from a database

HTTP methods are used to indicate the action an API client would like to perform on a given resource. Each HTTP method maps to a specific operation, such as creating, reading, updating, or deleting a resource, and an HTTP method must be included with every request to a REST API.

c. How does the HTTP request-response cycle work?

The HTTP request-response cycle is the foundation of communication between web clients (like browsers) and servers. A client initiates the cycle by sending an HTTP request, which the server then processes and responds with an HTTP response, completing the cycle. This process involves a client sending a request containing a request method (like GET or POST), the URL, and potentially additional information, and the server responding with a status code, headers, and the requested data.

The steps are:

1. **1. Client Initiates Request:**

A client, typically a web browser, sends an HTTP request to a specific server.

1. **2. Server Receives Request:**

The server receives the client's request and processes it, often by looking up the requested resource or performing other actions.

1. **3. Server Generates Response:**

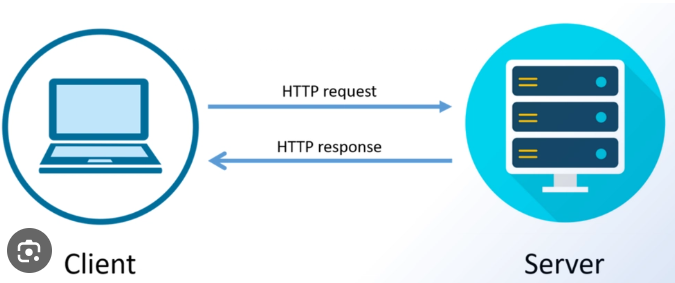
The server creates an HTTP response based on the request, which may include data, status codes (like 200 OK or 404 Not Found), headers, and potentially a body containing the requested information.

1. **4. Server Sends Response:**

The server sends the response back to the client.

1. **5. Client Receives and Displays Response:**

The client receives the response and renders it, displaying the data (like a webpage) to the user.



6.HTTPS vs HTTP

a. What is HTTPs, and how does it differ from HTTP?

HTTPS is the secure version of HTTP, designed to encrypt data transmission between a web browser and a website. HTTP, on the other hand, sends data in plain text, making it vulnerable to interception and modification.

Here's a more detailed comparison:

* **Security:**

HTTPS uses encryption (typically SSL/TLS) to protect data, ensuring it remains confidential and unreadable to eavesdroppers. HTTP transmits data in plain text, offering no such protection.

* **Verification:**

HTTPS verifies the server's identity using digital certificates, confirming the legitimate host of the website. HTTP does not have this verification, making it more susceptible to man-in-the-middle attacks.

* **Purpose:**

HTTPS is crucial for transmitting sensitive information like passwords or credit card details, where data privacy is paramount. HTTP is generally used for basic web browsing and less sensitive data.

* **Protocol:**

HTTPS is an extension of HTTP, incorporating encryption and verification. HTTP is the foundational protocol for web communication.

* **URL:**

HTTPS URLs begin with "https://" (indicating a secure connection), while HTTP URLs start with "http://".

* **Resources:**

HTTPS is slightly more resource-intensive due to encryption and decryption processes. HTTP is less resource-intensive.

* **Ports:**

HTTP typically operates on port 80, while HTTPS uses port 443.

b. Why is HTTPS important for modern web applications, and what role does SSL/TLS play?

HTTPS is crucial for modern web applications because it ensures secure and encrypted communication between a web browser and a server, protecting sensitive data from interception. SSL/TLS (Secure Sockets Layer/Transport Layer Security) is the technology that powers HTTPS, providing encryption and authentication for this secure communication.

Why HTTPS is Important:

* **Data Protection:**

HTTPS encrypts data transmitted between a client and a server, making it unreadable to anyone who might intercept it. This is crucial for protecting sensitive information like passwords, credit card details, and personal data.

* **Authentication:**

HTTPS verifies the identity of the website server, ensuring that the user is actually connecting to the legitimate site and not a malicious impersonation.

* **Integrity:**

HTTPS ensures that the data transmitted is not tampered with or modified during transit, preventing attackers from altering information.

* **Trust and Credibility:**

HTTPS is a sign of a website's commitment to security, building trust with users and potentially improving search engine rankings.

Role of SSL/TLS:

* **Encryption:**

SSL/TLS encrypts the data exchanged between a client and a server, making it unreadable without the proper decryption key.

* **Authentication:**

SSL/TLS establishes a secure connection by verifying the server's identity through a digital certificate.

* **Data Integrity:**

SSL/TLS uses cryptographic techniques to ensure that the data transmitted is not altered during transit.

* **HTTPS Enablement:**

SSL/TLS is the underlying technology that transforms HTTP into HTTPS, creating a secure connection for web applications.

7.HTTP Status Codes

a. What are HTTP status codes, and why are they important?

HTTP status codes are three-digit numerical responses sent by a web server to a client (like a browser) after a request is made. They indicate the outcome of the request, signaling whether it was successful, required further action, or resulted in an error. Understanding these codes is crucial for troubleshooting, improving user experience, and optimizing SEO.

Here's a breakdown:

* **1xx Informational:** The server acknowledges the request and is continuing processing.
* **2xx Success:** The request was successful, and the server has returned the requested data.
* **3xx Redirection:** The client needs to take further action, such as redirecting to a different URL.
* **4xx Client Error:** The request contains bad syntax or cannot be fulfilled due to an issue with the client.
* **5xx Server Error:** The server encountered an error while processing the request.

Why are they important?

* **Troubleshooting:**

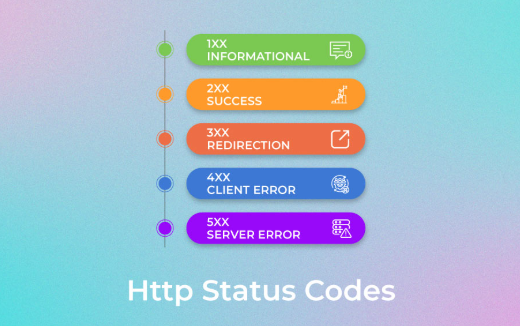
Status codes help identify issues with the website or network, allowing for faster debugging and resolution.

* **User Experience:**

Understanding status codes helps in providing better error handling and user-friendly experiences.

* **SEO:**

Search engines use status codes to determine the health and availability of a website, which impacts its ranking.



b. List and explain the meaning of the following HTTP status codes:200,301,404,500.

* **200 OK:** The request was successful, and the server has returned the requested content.
* **404 Not Found:** The server cannot find the requested resource.
* **301 Moved Permanently:** The requested resource has been permanently moved to a new URL.
* **500 Internal Server Error:** A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.