**Batch: T7**

**Practical No: 1**

**Title of Assignment: Study of Web and its Basics**

**Student Name: Shreeyash Shripad Dongarkar**

**Student PRN: 22510025**

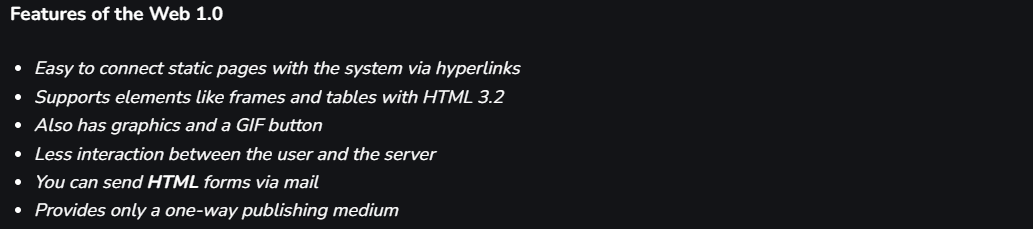
**Study and Describe Following Concepts in your words:**

**1) Evolution of Web including Web 3.0**

1. Web 1.0: The Static Web

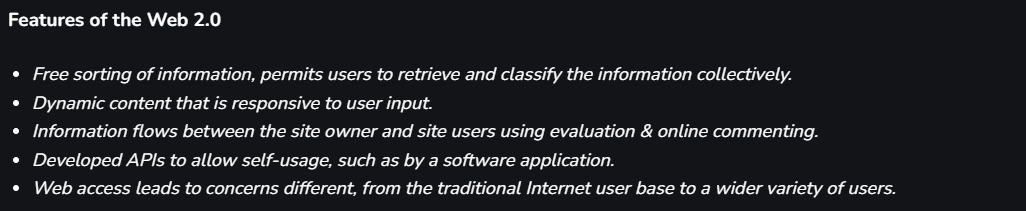
Web 1.0 refers to the first stage of the World Wide Web evolution. Earlier, there were only a few content creators in Web 1.0 with a huge majority of users who are consumers of content. Personal web pages were common, consisting mainly of static pages hosted on ISP-run**web servers**, or free **web hosting services**.

In Web 1.0 advertisements on websites while surfing the internet are banned. Also, in Web 1.0, **Ofoto** is an online digital photography website, on which users could store, share, view, and print digital pictures. Web 1.0 is a **content delivery network** (**CDN**) that enables the showcase of the piece of information on the websites. It can be used as a personal website. It costs the user as per pages viewed. It has directories that enable users to retrieve a particular piece of information. The era of Web 1.0 was roughly from 1991 to 2004.



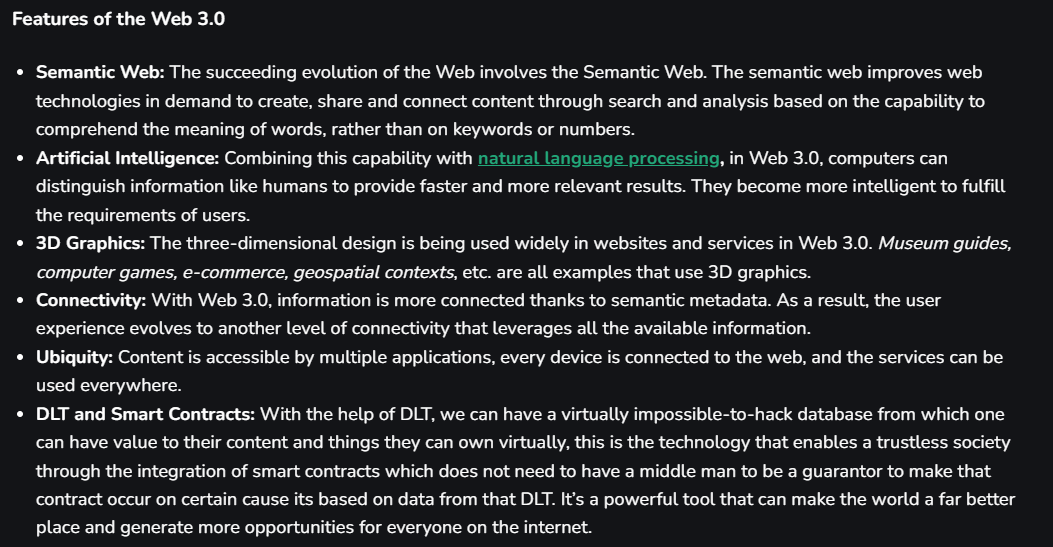
2. Web 2.0: The Social Web

2004 When the word Web 2.0 become famous due to the First Web 2.0 conference (later known as the Web 2.0 summit) held by Tim O’Reilly and Dale Dougherty, the term was coined by Darcy DiNucci in 1999. Web 2.0 refers to worldwide websites which highlight user-generated content, usability, and interoperability for end users.  Web 2.0 is also called the participative social web. It does not refer to a modification to any technical specification, but to modify the way Web pages are designed and used. The transition is beneficial but it does not seem that when the changes occur. Interaction and collaboration with each other are allowed by Web 2.0 in a social media dialogue as the creator of user-generated content in a virtual community. Web 2.0 is an enhanced version of Web 1.0.

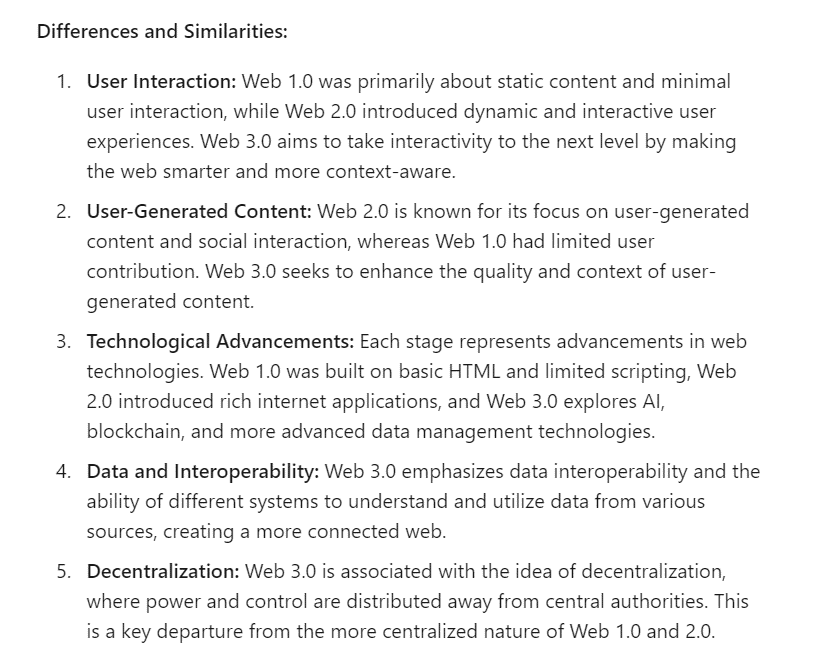
Web browser technologies are used in Web 2.0 development and it includes [**AJAX**](https://www.geeksforgeeks.org/ajax-introduction/)and JavaScript frameworks. 

3. Web 3.0: The Semantic Web

It refers to the evolution of web utilization and interaction which includes altering the Web into a database, with the integration of **DLT** (Distributed Ledger Technology **blockchain** is an example) and that data can help to make Smart Contracts based on the needs of the individual. It enables the up-gradation of the backend of the web, after a long time of focusing on the **frontend** (Web 2.0 has mainly been about AJAX, tagging, and other front-end user-experience innovation). Web 3.0 is a term that is used to describe many evolutions of web usage and interaction among several paths. In this, data isn’t owned but instead shared but still is, where services show different views for the same web / the same data.



Examples: 



**2) Which ports and protocols are used by web? Describe those in detail.**

**1. HTTP (HyperText Transfer Protocol)**

**Port:** 80

**Description:**

* **Purpose:** HTTP is the fundamental protocol used for transmitting data over the web. It is a request-response protocol where a client (usually a web browser) sends a request to a server, which then responds with the requested content.
* **Data Transmission:** HTTP transmits data in plain text, which means it is not encrypted. This makes it suitable for non-sensitive data but less secure compared to HTTPS.
* **Usage:** Commonly used for serving web pages and other resources like images, videos, and scripts.

**2. HTTPS (HyperText Transfer Protocol Secure)**

**Port:** 443

**Description:**

* **Purpose:** HTTPS is the secure version of HTTP. It uses encryption to protect data transmitted between the client and server, ensuring that sensitive information (like login credentials and payment details) is secure from eavesdropping and tampering.
* **Encryption:** HTTPS uses Transport Layer Security (TLS) or its predecessor, Secure Sockets Layer (SSL), to encrypt the data. This involves encrypting the entire communication channel to provide privacy and data integrity.
* **Usage:** HTTPS is essential for secure transactions and is increasingly used for all web traffic to ensure user privacy and data protection.

**3. FTP (File Transfer Protocol)**

**Port:** 21

**Description:**

* **Purpose:** FTP is used for transferring files between a client and a server over a network. It’s not limited to web servers but is often used for uploading and downloading files.
* **Data Transmission:** FTP does not encrypt data by default, making it less secure. Secure versions of FTP, like FTPS (FTP Secure) and SFTP (SSH File Transfer Protocol), are available for encrypted file transfers.
* **Usage:** Commonly used by web developers and administrators to upload website files to a web server.

**4. SMTP (Simple Mail Transfer Protocol)**

**Port:** 25 (unencrypted), 587 (encrypted)

**Description:**

* **Purpose:** SMTP is used for sending emails from a client to a mail server or between mail servers.
* **Data Transmission:** Standard SMTP does not encrypt messages, but it can be used with TLS (known as SMTPS) for secure email transmission on port 587.
* **Usage:** Commonly used by email clients and servers to send and relay emails.

**3) What is difference between HTTP and HTTPS?**

Encryption:

- HTTP: Does not use encryption. Data transmitted over HTTP is sent in plain text, making it vulnerable to interception and eavesdropping.

- HTTPS: Uses encryption via TLS (Transport Layer Security) or SSL (Secure Sockets Layer). This ensures that data is encrypted during transmission, protecting it from being read by unauthorized parties.

Data Security:

- HTTP: Lacks security features. Data can be intercepted and read by anyone with access to the network.

- HTTPS: Provides data security by encrypting the data. It also ensures data integrity, meaning the data cannot be altered during transmission without detection.

Authentication:

- HTTP: Does not provide authentication for the server. There is no assurance that the server the client is connecting to is legitimate.

- HTTPS: Includes server authentication through SSL/TLS certificates. This helps verify the identity of the server, providing assurance that the server is authentic.

Privacy:

- HTTP: Does not offer privacy. Data transmitted can be seen by anyone intercepting the traffic.

- HTTPS: Offers privacy by encrypting the data, making it unreadable to anyone intercepting the traffic.

Port:

- HTTP: Operates on port 80.

- HTTPS: Operates on port 443.

Performance:

- HTTP: Generally faster because it does not involve encryption overhead.

- HTTPS: May be slightly slower due to the encryption and decryption process, but the performance impact is minimal with modern hardware and optimizations.

Use Case:

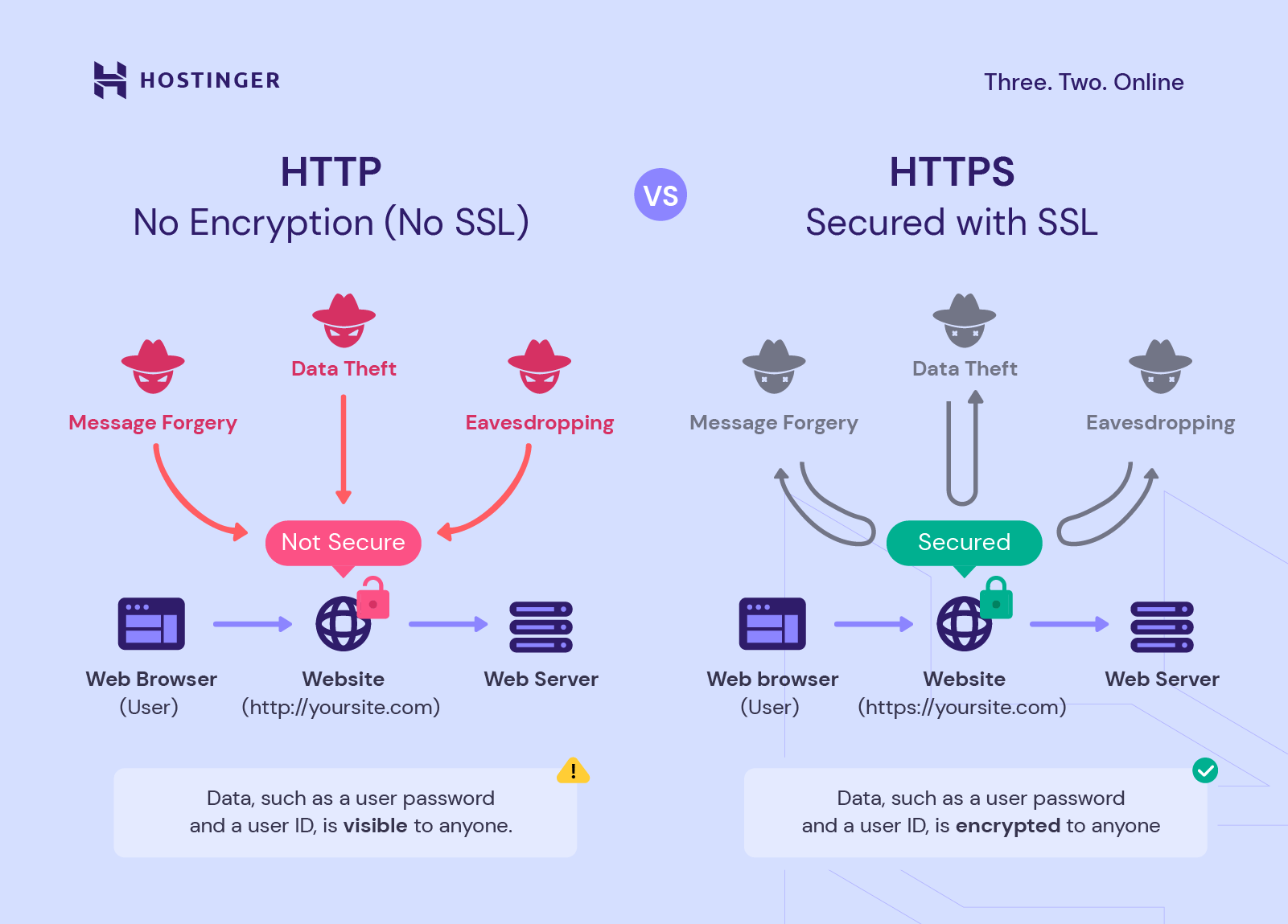
- HTTP: Suitable for non-sensitive information, such as public content where security is not a concern.

- HTTPS: Essential for handling sensitive information, such as login credentials, personal data, and financial transactions.

SEO Impact:

- HTTP: Less favourable in search engine rankings.

- HTTPS: Preferred by search engines like Google, contributing to better search engine optimization (SEO) and higher rankings.



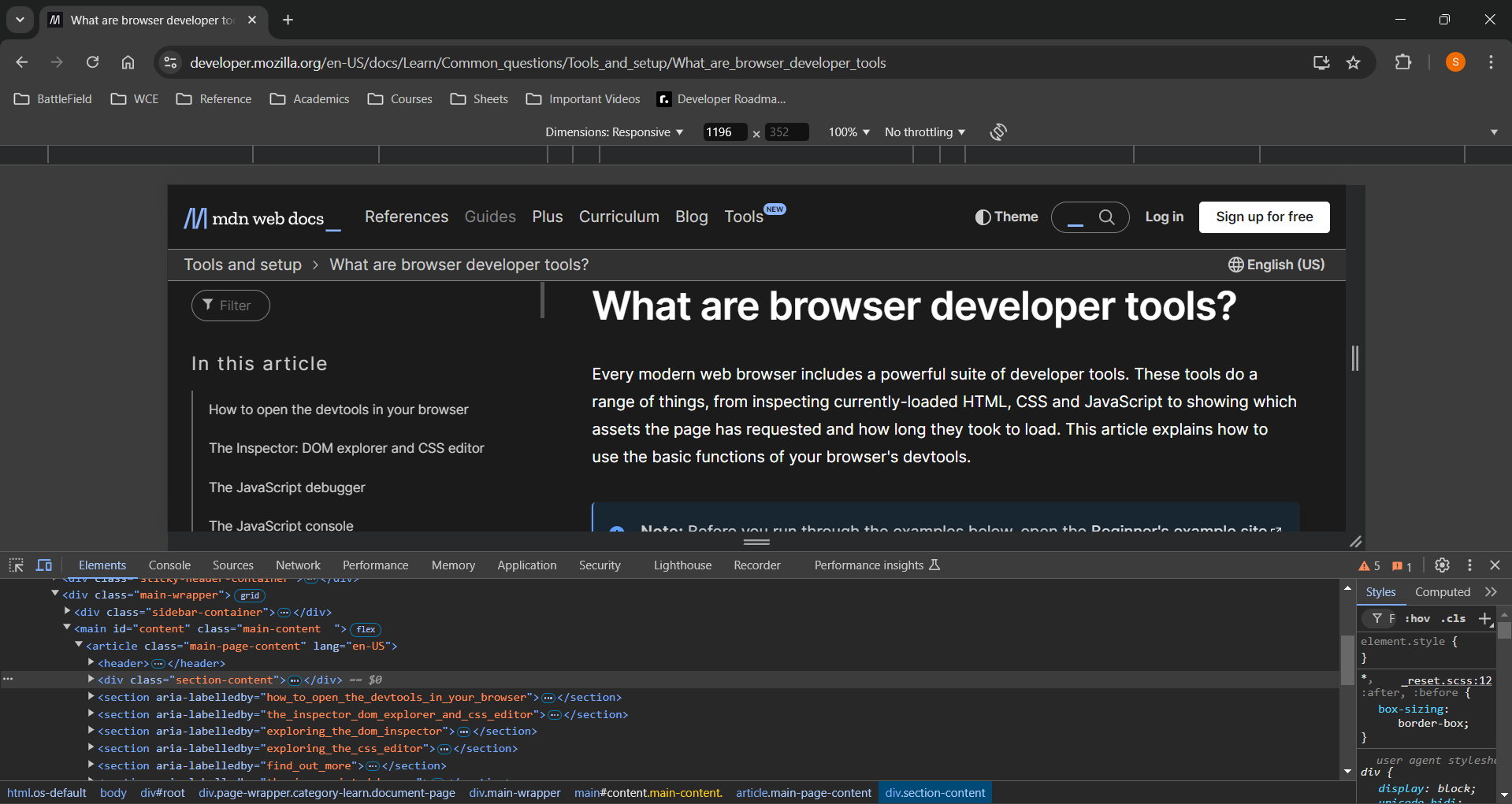
**4) What is Web Developer Tools and why it is needed?**

**Web Developer Tools** are built-in features in web browsers that help developers design, debug, and optimize websites. Here’s a brief overview:

* **Element Inspector:** View and modify HTML and CSS in real-time.
* **Console:** Log and debug JavaScript code and errors.
* **Network Monitor:** Track network requests and responses.
* **Sources Panel:** Debug and inspect source code.
* **Performance Monitor:** Analyse page load times and performance.
* **Memory Profiling:** Check for memory leaks and optimize usage.
* **Application Panel:** Manage client-side storage and data.
* **Security Panel:** View SSL/TLS certificate details and security status.
* **Responsive Design Mode:** Test how the site looks on different devices.

**Why They Are Needed:**

* **Debugging:** Identify and fix code issues.
* **Performance Optimization:** Improve page load times and responsiveness.
* **Design and Layout:** Modify and test page styles in real-time.
* **Testing:** Ensure compatibility across devices and browsers.
* **Security:** Check for vulnerabilities and ensure secure communication.
* **Efficiency:** Streamline the development process with integrated tools.



**5) Elaborate with diagram client server architecture and MVC architecture. When to use which architecture?**

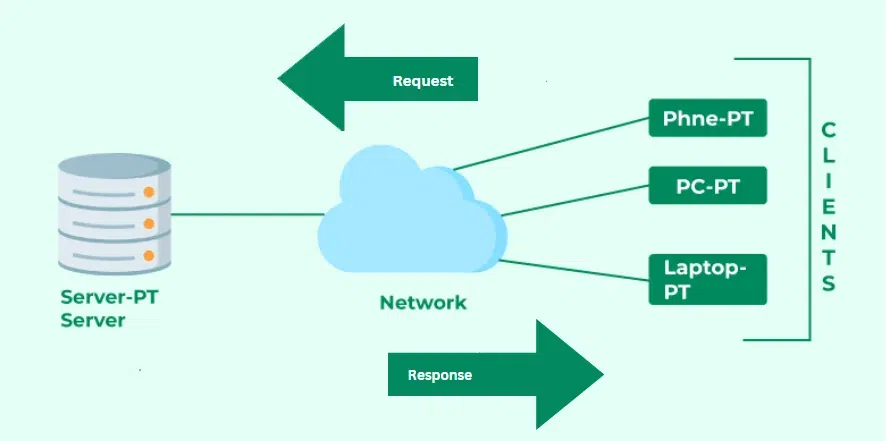
**Client-Server Model**

The Client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters called clients. In the client-server architecture, when the client computer sends a request for data to the server through the internet, the server accepts the requested process and delivers the data packets requested back to the client. Clients do not share any of their resources. Examples of the Client-Server Model are Email, World Wide Web, etc.

**How Does the Client-Server Model Work?**

* **Client:** When we say the word **Client**, it means to talk of a person or an organization using a particular service. Similarly in the digital world, a **Client** is a computer (**Host**) i.e. capable of receiving information or using a particular service from the service providers (**Servers**).
* **Servers:** Similarly, when we talk about the word **Servers**, It means a person or medium that serves something. Similarly in this digital world, a **Server** is a remote computer that provides information (data) or access to particular services.

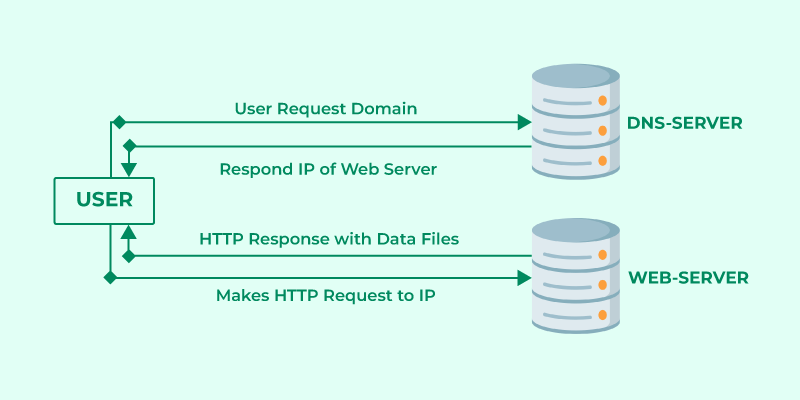
So, it is the **Client**requesting something and the **Server** serving it as long as it is in the database.



**How the Browser Interacts With the Servers?**

**There are a few steps to follow to interacts with the servers of a client.**

* User enters the **URL**(Uniform Resource Locator) of the website or file. The Browser then requests the [**DNS**(DOMAIN NAME SYSTEM)](https://www.geeksforgeeks.org/domain-name-system-dns-in-application-layer/) Server.
* **DNS Server** lookup for the address of the **WEB Server**.
* The**DNS Server** responds with the **IP address** of the **WEB Server**.
* The Browser sends over an **HTTP/HTTPS** request to the **WEB Server’s IP** (provided by the **DNS server**).
* The Server sends over the necessary files for the website.
* The Browser then renders the files and the website is displayed. This rendering is done with the help of **DOM** (Document Object Model) interpreter, **CSS** interpreter, and **JS Engine** collectively known as the **JIT** or (Just in Time) Compilers.



*Client Server Request and Response*

**Advantages of Client-Server Model**

* Centralized system with all data in a single place.
* Cost efficient requires less maintenance cost and Data recovery is possible.
* The capacity of the Client and Servers can be changed separately.

**Disadvantages of Client-Server Model**

* Clients are prone to viruses, Trojans, and worms if present in the Server or uploaded into the Server.
* Servers are prone to [Denial of Service (DOS)](https://www.geeksforgeeks.org/deniel-service-prevention/) attacks.
* Data packets may be spoofed or modified during transmission.
* Phishing or capturing login credentials or other useful information of the user are common and [MITM(Man in the Middle)](https://www.geeksforgeeks.org/mitm-man-in-the-middle-attack-using-arp-poisoning/) attacks are common.

MVC Architecture

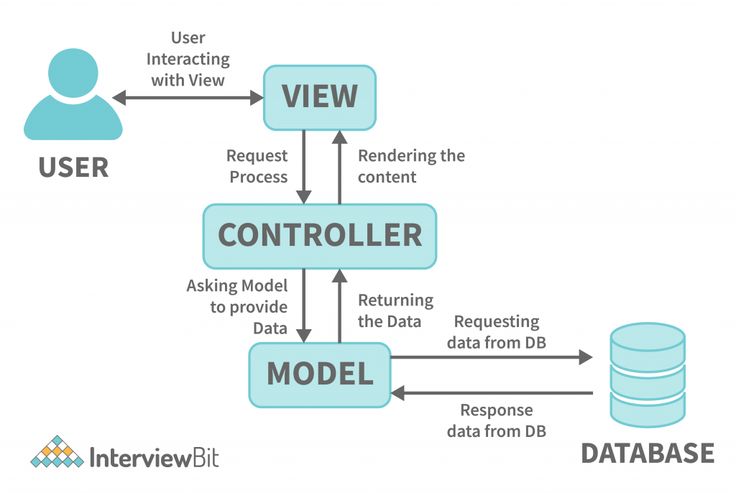
The [**Model-View-Controller (MVC)**](https://www.geeksforgeeks.org/mvc-design-pattern/) framework is an architectural/design pattern that separates an application into three main logical components **Model**, **View**, and **Controller**. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. It was traditionally used for desktop **graphical user interfaces (GUIs)**. Nowadays, MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. It is also used for designing mobile apps. The main goal of this design pattern was to solve the problem of users controlling a large and complex data set by splitting a large application into specific sections that all have their own purpose.

**Features of MVC**

* It provides a clear separation of **business logic, UI logic, and input logic.**
* It offers full control over your HTML and URLs which makes it easy to design web application architecture.
* It is a powerful URL-mapping component using which we can build applications that have comprehensible and searchable URLs.

**Components of MVC**

*MVC Architecture Design*



**Controller:**

The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary. The controller doesn’t have to worry about handling data logic, it just tells the model what to do. It processes all the business logic and incoming requests, manipulates data using the **Model**component, and interact with the **View**to render the final output.

**Responsibilities:**

* Receiving user input and interpreting it.
* Updating the Model based on user actions.
* Selecting and displaying the appropriate View.

**Example:**In a bookstore application, the Controller would handle actions such as searching for a book, adding a book to the cart, or checking out.

**View:**

The **View**component is used for all the UI logic of the application. It generates a user interface for the user. Views are created by the data which is collected by the model component but these data aren’t taken directly but through the controller. It only interacts with the controller.

**Responsibilities:**

* Rendering data to the user in a specific format.
* Displaying the user interface elements.
* Updating the display when the Model changes.

**Example:** In a bookstore application, the View would display the list of books, book details, and provide input fields for searching or filtering books.

**Model:**

The **Model**component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. It can add or retrieve data from the database. It responds to the controller’s request because the controller can’t interact with the database by itself. The model interacts with the database and gives the required data back to the controller.

**Responsibilities:**

* Managing data: CRUD (Create, Read, Update, Delete) operations.
* Enforcing business rules.
* Notifying the View and Controller of state changes.

**Example:**In a bookstore application, the Model would handle data related to books, such as the book title, author, price, and stock level.

**Advantages of MVC**

* Codes are easy to maintain and they can be extended easily.
* The MVC **model**component can be tested separately and developed simultaneously.
* It reduces complexity by dividing an application into three units.**Model, view, and controller.**
* It supports **Test Driven Development (TDD).**
* It works well for Web apps that are supported by large teams of web designers and developers.
* This architecture helps to test components independently as all classes and objects are independent of each other
* **Search Engine Optimization (SEO)** Friendly.

**Disadvantages of MVC**

* It is difficult to read, change, test, and reuse this model
* It is not suitable for building small applications.
* The inefficiency of data access in view.
* The framework navigation can be complex as it introduces new layers of abstraction which requires users to adapt to the decomposition criteria of MVC.
* Increased complexity and Inefficiency of data

Comparison and When to Use Which Architecture:

Client-Server Architecture:

Best for: Scenarios where a centralized system provides services to multiple clients, such as web services, APIs, and network-based applications.

Focus: Centralized resource management and service provision.

MVC Architecture:

Best for: Complex applications where clear separation of concerns between data, UI, and logic is required, such as web and desktop applications.

Focus: Organizing code into manageable components to improve maintainability and scalability.

**6) What is HTML and HTML5?**

HTML (HyperText Markup Language) is the standard language used for creating and structuring content on the web. It provides the basic building blocks for web pages and applications by defining elements like headings, paragraphs, links, images, and other content.

HTML (HyperText Markup Language)

Key Features:

- Basic Structure: HTML defines the structure of a web page using elements and tags. Tags like `<html>`, `<head>`, and `<body>` are used to organize and format content.

- Content Elements: HTML provides tags for various types of content, such as `<h1>` to `<h6>` for headings, `<p>` for paragraphs, `<a>` for links, `<img>` for images, and `<table>` for tables.

- Hyperlinks: HTML allows for the creation of hyperlinks using the `<a>` tag, enabling navigation between pages and websites.

- Forms: HTML includes form elements like `<input>`, `<textarea>`, and `<button>` for user interaction and data submission.

Usage:

- Web Pages: HTML is used to create the content and structure of web pages.

- Emails: HTML is also used in creating styled and formatted email content.

HTML5

Definition:

HTML5 is the fifth and latest major version of HTML. It introduces new features, elements, and APIs that enhance the capabilities of web pages and applications. HTML5 aims to provide better support for multimedia, improved performance, and enhanced semantics.

Key Features:

- New Semantic Elements: HTML5 introduces new tags like `<header>`, `<footer>`, `<article>`, and `<section>` to provide better structure and meaning to web content.

- Multimedia Support: HTML5 includes native support for audio and video through `<audio>` and `<video>` elements, eliminating the need for third-party plugins like Flash.

- Canvas and SVG: HTML5 provides the `<canvas>` element for drawing graphics and animations via JavaScript, and improved support for SVG (Scalable Vector Graphics).

- Form Enhancements: HTML5 introduces new form elements and attributes such as `<input type="date">`, `<input type="email">`, and `<input type="range">` for better form validation and user interaction.

- Local Storage: HTML5 offers local storage options through Web Storage API, allowing web applications to store data on the client side using `localStorage` and `sessionStorage`.

- Geolocation API: HTML5 provides the Geolocation API for accessing a user’s geographical location, which can be used for location-based services.

- Offline Capabilities: HTML5 includes the Application Cache (deprecated) and Service Workers (replacing it) to support offline functionality and improve web application performance.

**7) Which are the currently used versions of HTML and CSS for web development?**

As of now, the currently used versions for HTML and CSS in web development are:

HTML:

- HTML5: This is the latest and widely adopted version of HTML. It introduced new features and elements that support modern web development, including improved semantic elements, multimedia support, and APIs for enhanced functionality.

CSS:

- CSS3: This is the latest version of CSS that is widely used. CSS3 includes modules with new features such as advanced selectors, layout capabilities (e.g., Flexbox, Grid), animations, transitions, and responsive design techniques. CSS3 is modular, meaning new features and modules are added over time, so the term "CSS3" often refers to a collection of these modules rather than a single version.

Both HTML5 and CSS3 are extensively used in current web development to create modern, interactive, and responsive websites and applications.

**8) Which tools are available for front-end development and back-end development?**

Front-End Development Tools

- Text Editors/IDE: Visual Studio Code, Sublime Text, and Atom are popular choices for coding, offering features like syntax highlighting, extensions, and integrated terminal support.

- Browsers and DevTools: Chrome DevTools and Firefox Developer Tools provide essential features for debugging, performance analysis, and inspecting web pages in real-time.

- Frameworks/Libraries: React, Angular, and Vue.js help build dynamic user interfaces and single-page applications with reusable components and efficient data binding.

- CSS Preprocessors: Sass and Less add advanced features to CSS, such as variables and mixins, improving the efficiency and organization of stylesheets.

- Build Tools: Webpack, Gulp, and Parcel automate tasks like bundling, minifying, and transpiling code, streamlining the development workflow.

- Version Control: Git, along with platforms like GitHub, GitLab, and Bitbucket, is crucial for tracking code changes and collaborating with other developers.

- UI/UX Design: Figma, Adobe XD, and Sketch are used for designing user interfaces and creating interactive prototypes, enhancing the design and user experience.

- Package Managers: npm and Yarn manage project dependencies, allowing easy installation and updating of libraries and tools.

Back-End Development Tools

- Text Editors/IDE: Visual Studio Code, IntelliJ IDEA, and PyCharm are commonly used for server-side coding, offering advanced features for debugging and code management.

- Languages: Node.js, Python, Java, PHP, and Ruby are popular server-side languages, each with its own strengths and frameworks for building web applications.

- Frameworks/Libraries: Express.js (for Node.js), Django (for Python), Spring Boot (for Java), Laravel (for PHP), and Flask (for Python) provide robust structures for developing server-side logic and APIs.

- Databases: MySQL, PostgreSQL, MongoDB, and SQLite are widely used databases for storing and managing application data, each offering different features and data models.

- APIs/Web Services: REST and GraphQL are methods for building and interacting with APIs, enabling data exchange between client and server applications.

- Version Control: Git, supported by platforms like GitHub, GitLab, and Bitbucket, helps manage code changes and coordinate development among team members.

- Containerization: Docker and Kubernetes streamline the deployment and management of applications by encapsulating them in containers and orchestrating their deployment.

- Deployment/Hosting: Heroku, AWS, Azure, and Netlify/Vercel are platforms for deploying, hosting, and scaling web applications, providing various services and infrastructure solutions.

**9) What MERN stack includes? Why and when it is preferred for web development?**

The MERN stack includes:

- MongoDB: NoSQL database for flexible data storage.

- Express.js: Web framework for Node.js to handle server-side logic and routing.

- React: JavaScript library for building dynamic user interfaces.

- Node.js: JavaScript runtime for executing server-side code.

Why and When MERN Stack is Preferred:

- Unified Language: Uses JavaScript across the entire stack, simplifying development.

- Scalability: Handles high traffic and large data volumes well.

- SPA Development: React excels at creating single-page applications.

- Performance: Offers high performance with non-blocking I/O and virtual DOM.

- Community Support: Strong ecosystem and resources for rapid development.

**10) List out newly introduced input types, APIs, form elements, and elements that support media content in HTML5.**

Newly Introduced HTML5 Features

Input Types:

1. <input type="date"> - Allows users to select a date.

2. <input type="time"> - Allows users to select a time.

3. <input type="datetime-local"> - Allows users to select both date and time.

4. <input type="month"> - Allows users to select a month and year.

5. <input type="week"> - Allows users to select a week and year.

6. <input type="email"> - Validates email addresses.

7. <input type="url"> - Validates URLs.

8. <input type="number"> - Allows numeric input with optional constraints like min, max, and step.

9. <input type="range"> - Provides a slider for selecting a value within a range.

10. <input type="color"> - Opens a color picker for selecting a color.

APIs:

1. Geolocation API: Allows websites to access a user's geographical location.

2. Web Storage API: Provides localStorage and sessionStorage for storing data on the client-side.

3. Web Workers API: Allows background processing in web applications to improve performance.

4. Canvas API: Provides a <canvas> element for drawing graphics, animations, and visualizations.

5. WebSocket API: Enables full-duplex communication channels over a single TCP connection.

6. Fetch API: Provides a modern way to make network requests, replacing XMLHttpRequest.

7. WebRTC API: Facilitates peer-to-peer audio, video, and data communication directly between browsers.

Form Elements:

1. <datalist> - Provides a list of predefined options for an <input> element.

2. <output> - Represents the result of a calculation or user action.

3. <progress> - Displays the progress of a task.

4. <meter> - Represents a measurement within a known range, such as disk usage or battery level.

Elements Supporting Media Content:

1. <audio> - Embeds audio content with built-in controls for playback.

2. <video> - Embeds video content with controls for playback, and supports various video formats.

3. <source> - Provides multiple media sources for <audio> and <video> elements, allowing the browser to choose the best format.

4. <track> - Provides text tracks for <video> and <audio> elements, such as subtitles or captions.

These new features and elements in HTML5 enhance the capabilities of web forms, media handling, and client-side interaction, leading to richer and more interactive web applications.

**11) Explain HTML5 Web storage.**

HTML5 Web Storage provides a way to store data on the client side directly within the user's browser. It includes two primary storage mechanisms:

1. localStorage

- Scope: Data stored in `localStorage` persists across browser sessions. It remains available even when the browser is closed and reopened, and is accessible from different tabs and windows of the same origin.

- Capacity: Typically allows for around 5-10MB of data storage, varying by browser.

- Use Cases: Ideal for storing data that needs to be preserved between sessions, such as user preferences or settings.

2. sessionStorage

- Scope: Data stored in `sessionStorage` is available only for the duration of the page session. It is retained as long as the browser tab or window is open. Once the tab or window is closed, the data is deleted.

- Capacity: Also typically allows for around 5-10MB of storage.

- Use Cases: Suitable for temporary data relevant only to the current session, such as form data or session-specific settings.

Key Points:

- Storage Limits: Web Storage offers larger storage capacity compared to cookies, which are limited to about 4KB.

- Security: Data stored is accessible only from the same origin (same protocol, domain, and port). It is not sent with every HTTP request like cookies.

- Data Handling: Web Storage handles data as strings. Complex data types must be converted to and from strings (e.g., JSON format).

Benefits:

- Convenience: Easy to use for storing and retrieving data without needing server interaction.

- Performance: Reduces server load by keeping data on the client side.

- Persistence: `localStorage` provides data persistence that lasts beyond browser sessions.

Limitations:

- Security: Data is not encrypted and can be accessed by JavaScript running on the same origin.

- Capacity Constraints: Though larger than cookies, the storage is still limited compared to server-side databases.

HTML5 Web Storage is useful for maintaining state and user data in web applications, enhancing the user experience by allowing for efficient, client-side data management.

**Problem Statement: Create a static web page for “Portfolio” of your own which will include Photo, Name, Class, College Name, Achievements/ Certificates, Extracurricular Activities, Courses Completed, hobbies, technical expertise, etc.**

**Technologies Used: HTML and CSS**

Developed a responsive web page using semantic HTML5 to structure the content. Leveraged advanced CSS concepts, including Flexbox and CSS Grid, for creating complex layouts that adapt seamlessly to various screen sizes.

