SECTION A

1)Write the history of internet.

The internet's history began in the 1960s with ARPANET, a project funded by the U.S. Department of Defense to create a decentralized communication network. In 1983, ARPANET adopted the TCP/IP protocol, forming the basis of the modern internet. The 1990s saw the rise of the World Wide Web, invented by Tim Berners-Lee, making the internet more accessible to the public. By the mid-1990s, the internet became commercialized, leading to its rapid expansion and integration into daily life.

2) What is News-group?

A newsgroup is an online discussion forum where users post messages about specific topics. Organized within the Usenet system, newsgroups function similarly to bulletin boards, allowing individuals to share information, ask questions, and participate in conversations. Each newsgroup focuses on a particular subject, enabling targeted discussions. Messages, or "articles," are stored on servers and can be read and responded to by others, fostering a collaborative exchange of ideas and information.

3)Define HTML.

HyperText Markup Language is the standard markup language for documents designed to be displayed in a web browser. It defines the content and structure of web content. It is often assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

4)What is web portal?

A web portal is a specially designed website that brings together information from diverse sources in a unified way. Typically, each information source gets its dedicated area on the page for displaying information. Web portals often provide personalized content, user interfaces, and services such as email, forums, and search engines. They serve as entry points to a wide range of resources and services, simplifying user access to various types of data and applications on the internet.

5)Define image maps.

Image maps refer to a feature in web design where different areas of an image can be defined as clickable links, each leading to a different URL or action. This allows designers to create interactive graphics where specific regions or hotspots on an image can trigger navigation to various destinations, enhancing user engagement and interactivity on websites. Image maps are commonly used for navigation menus, diagrams, and other graphical interfaces on the web.

6)Why JavaScript is used?

JavaScript is used to create interactive and dynamic web pages. It enables developers to enhance user experience by allowing real-time content updates, form validations, animations, and interactive elements like sliders and pop-ups. JavaScript also facilitates client-side scripting, reducing server load and improving performance. With frameworks and libraries like React, Angular, and Vue.js, JavaScript plays a crucial role in developing modern web applications, ensuring responsiveness and an engaging user interface.

7.)What is DHTML?

DHTML (Dynamic HyperText Markup Language) refers to a collection of technologies used together to create interactive and animated web content. It combines HTML, CSS, and JavaScript to enable dynamic changes to web pages without requiring a reload. With DHTML, elements on a web page can move, change, and respond to user actions in real-time, enhancing the user experience by making web pages more responsive and engaging.

8.)What is Dreamweaver in web technologies?
Dreamweaver is a web development application originally developed by Macromedia and later acquired by Adobe. It is used by web developers and designers to create, edit, and manage websites and web applications. Dreamweaver supports various programming languages like HTML, CSS, JavaScript, and server-side scripting languages such as PHP and ASP. It provides a visual interface for designing web pages and also includes features for code editing, syntax highlighting, FTP integration, and site management, making it a versatile tool for both beginners and experienced web developers.

9.)Write the features of markup languages. 80 word limit
Markup languages define structure and formatting of text and data in digital documents. Key features include simplicity, extensibility through tags, platform independence, and human readability. They facilitate document interpretation by computers and humans alike, supporting content structure with tags like HTML's <p> for paragraphs. Markup languages ensure consistent presentation across devices and browsers while enabling semantic meaning through tags. They're essential for web content, e-books, and document formatting, catering to diverse needs from basic text to complex data structures in various digital environments.

10)Define the Frames and floating frames.
Frames in web design allow multiple HTML documents to be displayed in a single web browser window. They divide the browser window into multiple rectangular sections, each containing a separate HTML document. Frames facilitate creating layouts with fixed regions for navigation, content, and headers.

Floating frames, also known as iframe (inline frame), are a type of frame that allows embedding one HTML document within another. Unlike traditional frames, iframes can be positioned anywhere within a webpage and are more flexible for integrating external content like advertisements or interactive elements.

SECTION B

11)Difference between HTML and XML.
the key differences between HTML and XML:

Feature	HTML	XML
Purpose	Defines structure and presentation of web pages	Defines structure and data for documents
Syntax	Contains predefined tags for formatting	Requires user-defined tags and structure
Extensibility	Limited, as tags are predefined	Highly extensible, allowing custom tags
Validation	Limited validation against DTD or schema	Can be validated against a DTD or XML schema
Tag Usage	Tags have predefined meanings (e.g., <p>,)	Tags are defined based on document content
Usage	Used for creating web pages	Used for data storage, interchange, and structured documents
Semantic Meaning	Focuses on presentation and structure	Focuses on data representation and structure

HTML (HyperText Markup Language) primarily focuses on defining the structure and presentation of web pages, using predefined tags for formatting content. XML (eXtensible Markup Language), on the other hand, allows users to define their own tags and structure,

making it highly customizable for data representation and interchange between different systems.

12)Describe the Email and its protocol with Diagram.

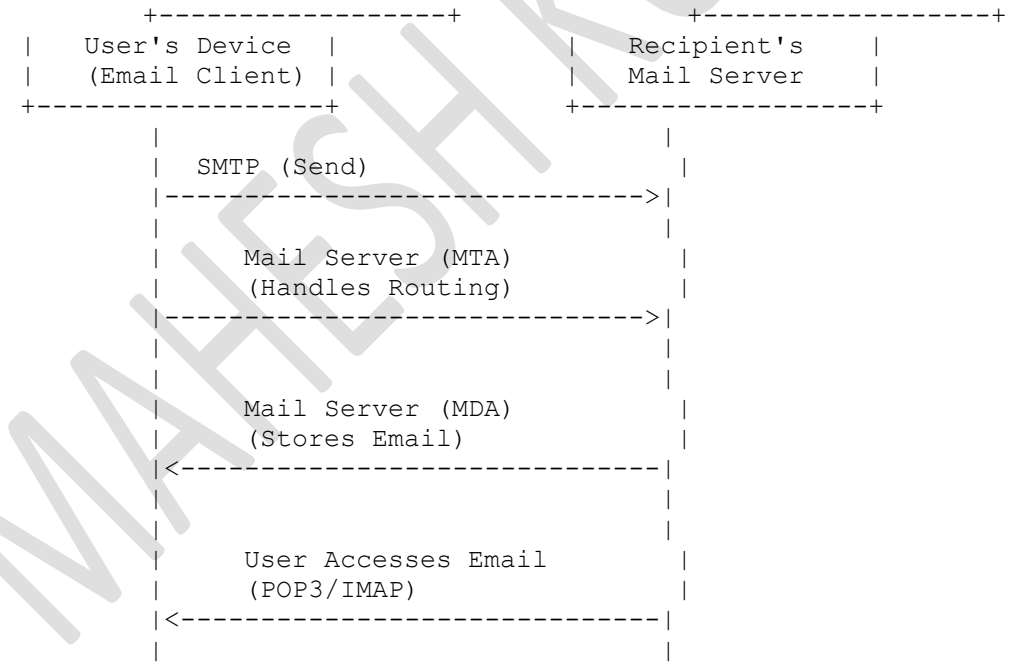
Email and its Protocol:Email (Electronic Mail) is a method of exchanging digital messages between users over the internet or other computer networks. It allows individuals and businesses to communicate asynchronously, sending text, files, and multimedia content.

Components of Email:

1. **User Agent (Email Client)**: Software used by the user to compose, send, receive, and manage emails (e.g., Outlook, Gmail).
2. **Mail Server**: Handles storage, retrieval, and forwarding of emails. Consists of:
   - **Mail Transfer Agent (MTA)**: Routes emails between servers using SMTP (Simple Mail Transfer Protocol).
   - **Mail Delivery Agent (MDA)**: Stores incoming emails in a user's mailbox.
   - **Mail Access Protocols**: POP3 (Post Office Protocol) or IMAP (Internet Message Access Protocol) for retrieving emails from the server.

Diagram:

Email and its protocol with Diagram.



13)What are the different data types present in JavaScript?

JavaScript supports several data types, categorized into two main categories: primitive and non-primitive (object) types. Here are the primary data types in JavaScript:

1. **Primitive Data Types:**
   - **Number:** Represents numeric values, including integers and floating-point numbers.
   - **String:** Represents sequences of characters, enclosed in single or double quotes.
   - **Boolean:** Represents logical values `true` and `false`.

- **Undefined:** Represents a variable that has been declared but not assigned a value.
- **Null:** Represents the intentional absence of any object value.
- **Symbol:** Represents unique identifiers (introduced in ES6).

2. **Non-primitive Data Type (Object):**
   - **Object:** Represents a collection of key-value pairs and is used for more complex data structures. Arrays, functions, and objects themselves are all types of objects in JavaScript.

14)Why do we use the word "debugger" in JavaScript? 120 word .with real life example
The term "debugger" in JavaScript refers to tools or software designed to help programmers identify and fix errors, known as bugs, in their code. It enables developers to step through code execution, inspect variables, and track program flow to pinpoint where issues occur. The term originates from an actual insect found inside a malfunctioning early computer, which led engineers to discover a hardware bug causing the issue. For example, a web developer might use a debugger like Chrome DevTools to trace through JavaScript functions to find why a form validation script isn't working as expected, ensuring the application functions correctly before deployment.

15)Describe all Operators in JavaScript.
JavaScript includes various types of operators that allow developers to perform operations on variables and values. Here's an overview of the types of operators in JavaScript:

1. **Arithmetic Operators:** Perform arithmetic operations on numeric values. Examples include `+` (addition), `-` (subtraction), `*` (multiplication), `/` (division), and `%` (modulus).
2. **Assignment Operators:** Assign values to variables. Examples include `=` (assign), `+=` (add and assign), `-=` (subtract and assign), `*=` (multiply and assign), `/=` (divide and assign), and `%=` (modulus and assign).
3. **Comparison Operators:** Compare two values and return a Boolean result (`true` or `false`). Examples include `==` (equal to), `!=` (not equal to), `===` (strict equal to), `!==` (strict not equal to), `>` (greater than), `<` (less than), `>=` (greater than or equal to), and `<=` (less than or equal to).
4. **Logical Operators:** Combine Boolean expressions and return a Boolean result. Examples include `&&` (logical AND), `||` (logical OR), and `!` (logical NOT).
5. **Bitwise Operators:** Perform operations on binary representations of numbers. Examples include `&` (bitwise AND), `|` (bitwise OR), `^` (bitwise XOR), `<<` (left shift), `>>` (right shift), and `>>>` (unsigned right shift).
6. **Unary Operators:** Operate on a single operand. Examples include `++` (increment), `--` (decrement), `-` (unary negation), and `+` (unary plus).
7. **Conditional (ternary) Operator:** Provides a shorthand way to write conditional statements. Example: `condition ? expr1 : expr2`.
8. **String Operators:** Concatenate strings using the `+` operator.
9. **Type Operators:** Check the data type of a variable. Example: `typeof` operator.

16)Discuss what are CSS and its types?
CSS (Cascading Style Sheets) is a language used to describe the presentation of HTML (and XML) documents. It defines how elements are displayed on a web page, including their layout, colors, fonts, and other stylistic aspects. CSS separates content from presentation, allowing developers to create consistent and visually appealing web pages across different devices and screen sizes.

Types of CSS:

1. **Inline CSS:** Applied directly to an HTML element using the `style` attribute within the element's tag. Example: `<p style="color: red;">`.
2. **Internal (Embedded) CSS:** Defined within the `<style>` element in the `<head>` section of an HTML document. Example:

```
<style>

    body {
        font-family: Arial, sans-serif;
        background-color: #f0f0f0;
    }
</style>
```

3. **External CSS:** Linked to an HTML document using the `<link>` element in the `<head>` section. The CSS rules are stored in a separate `.css` file. Example:

```
<link rel="stylesheet" href="styles.css">
```

Where `styles.css` contains CSS rules like:

```
css
Copy code
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
}
```

17)What is an object in JavaScript?

In JavaScript, an object is a standalone entity, grouping data and functionality (methods) into a single unit. It consists of properties (key-value pairs) where each key is a string (or Symbol) and each value can be any data type, including other objects, functions, or primitive values. Objects in JavaScript are instances of predefined or custom-defined classes.

For example, consider a `person` object:

```
let person = {
    firstName: 'John',
    lastName: 'Doe',
    age: 30,
    fullName: function() {
       return this.firstName + ' ' + this.lastName;
    }
};
```

Here, `person` is an object with properties (`firstName`, `lastName`, `age`) and a method (`fullName`). Objects allow for structured organization of data and behaviors, facilitating complex data manipulation and interaction within JavaScript programs.

SECTION –C

18)Write code and diagrams to create any type of home page in HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home Page</title>
    <link rel="stylesheet" href="styles.css"> <!-- External CSS file for
styling -->
</head>
<body>
    <header>
        <div class="logo">
            <h1>Company Name</h1>
        </div>
        <nav>
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">About</a></li>
                <li><a href="#">Services</a></li>
                <li><a href="#">Contact</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section class="hero">
            <h2>Welcome to Our Website!</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Nullam sed ante id turpis consequat euismod.</p>
            <a href="#" class="btn">Learn More</a>
        </section>

        <section class="services">
            <h3>Our Services</h3>
            <ul>
                <li>Service 1</li>
                <li>Service 2</li>
                <li>Service 3</li>
            </ul>
        </section>
    </main>

    <footer>
        <p>&copy; 2024 Company Name. All rights reserved.</p>
    </footer>
</body>
</html>
```

CSS Code (`styles.css`):

```
/* Basic styles for layout */

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

header {
    background-color: #333;
    color: #fff;
    padding: 10px 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}
```

```
.logo h1 {
    margin: 0;
}

nav ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

nav ul li {
    display: inline;
    margin-right: 20px;
}

nav ul li a {
    color: #fff;
    text-decoration: none;
}

main {
    padding: 20px;
}

.hero {
    background-color: #f0f0f0;
    padding: 20px;
    text-align: center;
}

.btn {
    display: inline-block;
    background-color: #333;
    color: #fff;
    padding: 10px 20px;
    text-decoration: none;
    border-radius: 5px;
}

.btn:hover {
    background-color: #555;
}

.services {
    margin-top: 20px;
}

footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px 0;
    position: absolute;
    bottom: 0;
    width: 100%;
}
```

Diagram:

```
-------------------------------------------------------------
|                          HEADER                           |
|-----------------------------------------------------------|
| Logo | Navigation                                         |
|-----------------------------------------------------------|
```

```
|                          MAIN                                |
|--------------------------------------------------------------|
| Hero Section                                                 |
|--------------------------------------------------------------|
| Services Section                                             |
|--------------------------------------------------------------|
|                          FOOTER                              |
|--------------------------------------------------------------|
| Copyright                                                    |
----------------------------------------------------------------
```

19)Differentiate between an ordered list and an unordered list in HTML.

Feature	Ordered List (`<ol>`)	Unordered List (`<ul>`)
Definition	Represents a list of items where each item is sequentially numbered (1, 2, 3, etc.).	Represents a list of items where each item is bulleted with a bullet point or other marker.
Markup Example	```html - Item 1 - Item 2 - Item 3 	```html - Item 1 - Item 2 - Item 3
Display	Automatically numbers each item in sequential order.	Displays each item with a bullet point or other marker (e.g., disc, square).
Attributes	Supports the `type` attribute to define numbering style: - `type="1"`: Numbers (default). - `type="A"`: Uppercase letters (A, B, C, ...). - `type="a"`: Lowercase letters (a, b, c, ...). - `type="I"`: Uppercase Roman numerals (I, II, III, ...). - `type="i"`: Lowercase Roman numerals (i, ii, iii, ...).	Supports the `type` attribute to define bullet style: - `type="disc"`: Filled circle (default). - `type="circle"`: Hollow circle. - `type="square"`: Filled square.
Accessibility	Useful for lists that require a specific order or sequence.	Ideal for lists where order is not important or for presenting options.
Usage	Often used for steps, procedures, rankings, or any ordered sequence.	Commonly used for menus, lists of items, options, or where sequence isn't significant.
Semantics	Provides semantic meaning and structure to ordered information.	Organizes information into discrete, non-sequential items.

20)Write code and diagrams to create any type of Contact page in HTML.
HTML Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Contact Us</title>
    <link rel="stylesheet" href="styles.css"> <!-- External CSS file for
styling -->
</head>
<body>
    <header>
        <h1>Contact Us</h1>
        <nav>
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="about.html">About</a></li>
                <li><a href="services.html">Services</a></li>
                <li><a href="contact.html" class="active">Contact</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section class="contact-form">
            <h2>Send us a Message</h2>
            <form action="submit_form.php" method="POST">
                <div class="form-group">
                    <label for="name">Your Name:</label>
                    <input type="text" id="name" name="name" required>
                </div>
                <div class="form-group">
                    <label for="email">Your Email:</label>
                    <input type="email" id="email" name="email" required>
                </div>
                <div class="form-group">
                    <label for="message">Message:</label>
                    <textarea id="message" name="message" rows="4"
required></textarea>
                </div>
                <button type="submit">Send Message</button>
            </form>
        </section>
    </main>

    <footer>
        <p>&copy; 2024 Company Name. All rights reserved.</p>
    </footer>
</body>
</html>
```

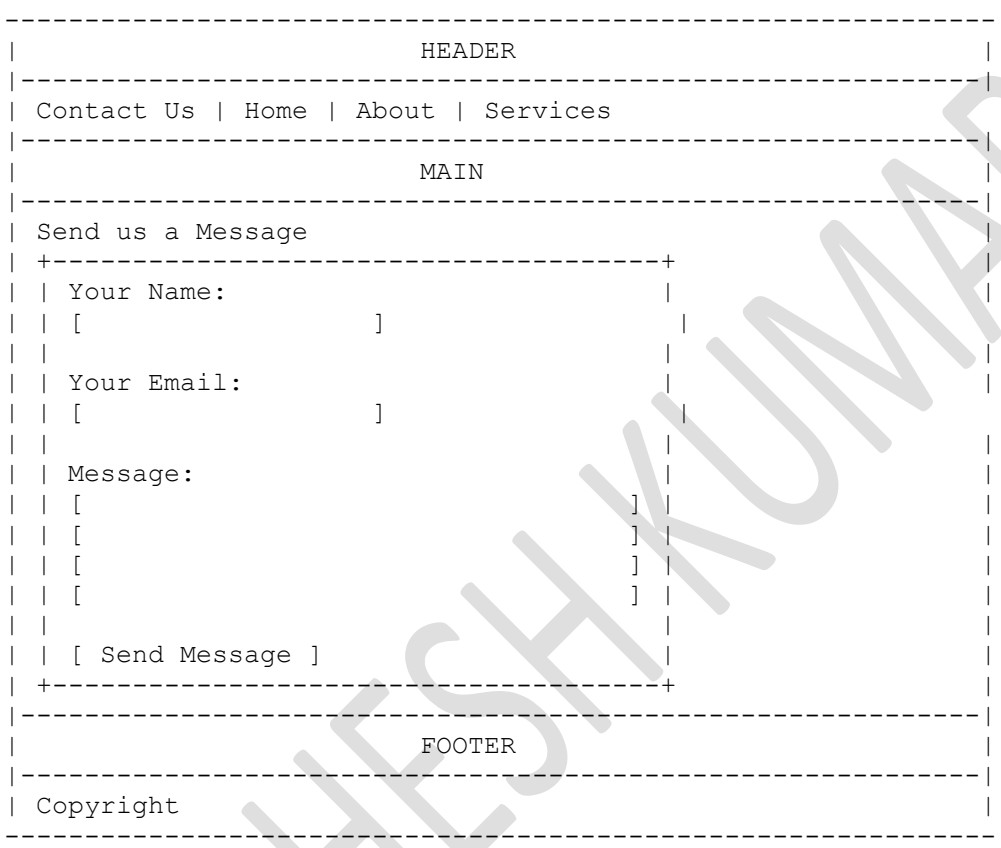
CSS Code (`styles.css`):

```
/* Basic styles for layout */


body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

header {
    background-color: #333;
    color: #fff;
    padding: 10px 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
```

```
}

nav ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

nav ul li {
    display: inline;
    margin-right: 20px;
}

nav ul li a {
    color: #fff;
    text-decoration: none;
}

main {
    padding: 20px;
}

.contact-form {
    max-width: 600px;
    margin: 20px auto;
    background-color: #f0f0f0;
    padding: 20px;
    border-radius: 5px;
}

.contact-form h2 {
    margin-top: 0;
}

.form-group {
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
}

input[type="text"],
input[type="email"],
textarea {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

button[type="submit"] {
    background-color: #333;
    color: #fff;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button[type="submit"]:hover {
    background-color: #555;
}
```

```
footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px 0;
    position: fixed;
    bottom: 0;
    width: 100%;
}
```

Diagram:



21)Explain about the purpose of DTD.

The purpose of a Document Type Definition (DTD) in HTML and XML is to define the structure, elements, and rules that govern the markup language. DTDs serve several important purposes:

1. **Define Document Structure:** DTDs specify which elements can appear in a document, their order, nesting rules, and attributes they can have. This ensures consistency and structure across documents.
2. **Validation:** DTDs allow documents to be validated against a specific set of rules. Validation ensures that the document adheres to the specified structure and format, reducing errors and ensuring data integrity.
3. **Interoperability:** By defining a standard structure, DTDs promote interoperability between different systems and applications. They ensure that documents created by one system can be correctly interpreted and processed by others.
4. **Document Type Identification:** DTDs help identify the type and version of a document. This information is crucial for applications and parsers to correctly interpret and handle the document content.

5. **Error Checking:** During document creation or editing, DTDs help detect errors such as missing or incorrectly formatted elements and attributes. This aids in maintaining document quality and consistency.
6. **Document Documentation:** DTDs serve as documentation that describes the elements, attributes, and rules used in a specific markup language. This documentation is invaluable for developers and users working with the documents.

Overall, DTDs play a vital role in defining, validating, and ensuring the proper structure and integrity of documents in HTML and XML, thereby facilitating efficient data exchange and consistent document processing across different platforms and applications.

22)Differentiate between client side scripting and server side scripting.in table form

Feature	Client-Side Scripting	Server-Side Scripting
Execution Location	Executes on the user's web browser (client side).	Executes on the web server (server side).
Code Visibility	Code is visible to users and can be viewed/modified in browser developer tools.	Code is not visible to users; only the output is sent to the client.
Processing	Requires client resources (CPU, memory) to execute.	Executes on the server, utilizing server resources (CPU, memory).
Browser Dependency	Execution depends on browser capabilities and settings.	Execution is independent of client browser capabilities.
Performance	Faster initial page rendering since all processing is done locally.	Slower initial page rendering due to server request/response cycle.
Security	More vulnerable to security threats such as XSS (Cross-Site Scripting).	Less vulnerable to client-side attacks as code is not exposed.
Interaction with User	Enables immediate interaction and responsiveness without server interaction.	Interaction requires server communication for data processing.
Examples	JavaScript (in-browser scripting), client-side frameworks (React, Angular).	PHP, Python (Django), Ruby (Rails), ASP.NET, Node.js.
Common Use Cases	Form validation, dynamic content updates (AJAX), client-side animations.	Database operations, user authentication, dynamic content generation.
Scalability	Limited by client resources and browser capabilities.	Scalable depending on server capacity and configuration.

Summary:

- **Client-Side Scripting**: Executes in the user's browser, visible and accessible locally, suitable for interactive and responsive features but may expose vulnerabilities.
- **Server-Side Scripting**: Executes on the server, hidden from users, handles server-side operations securely but with slower initial rendering due to server processing.