**Expt 1 - Website Using HTML**

\*\*Introduction:\*\*

Welcome to our website, where we showcase the versatility and power of HTML through the demonstration of essential tags. HTML, the backbone of the web, empowers developers to structure and present content seamlessly across various platforms. In this demonstration, we explore five fundamental HTML tags—Headings, Paragraphs, Images, Anchors, and Tables—each playing a crucial role in web development.

\*\*Description of HTML Tags:\*\*

1. \*\*Headings:\*\* HTML offers six levels of headings, ranging from `<h1>` to `<h6>`, allowing developers to organize content hierarchically and emphasize key points effectively.

2. \*\*Paragraphs:\*\* Enclosed within `<p>` tags, paragraphs facilitate the presentation of textual information, ensuring clarity and readability for website visitors.

3. \*\*Images:\*\* The `<img>` tag enables the integration of visual elements such as graphics, illustrations, and photographs, enhancing the visual appeal and context of web pages.

4. \*\*Anchors:\*\* Represented by the `<a>` tag, anchors create hyperlinks to other web pages or resources, enabling seamless navigation and fostering user interaction.

5. \*\*Tables:\*\* Defined by a combination of `<table>`, `<tr>`, `<th>`, and `<td>` tags, tables provide a structured layout for presenting tabular data, facilitating comparison and analysis.

\*\*Conclusion:\*\*

By mastering these fundamental HTML tags, developers can create dynamic and engaging websites that deliver compelling content and seamless user experiences. Explore our demonstration to learn how these tags can elevate your web development projects.

**Expt 2 - Website Using HTML and CSS**

\*\*Theoretical Background:\*\*

Cascading Style Sheets (CSS) is a fundamental technology used in web development to define the presentation and layout of HTML documents. With CSS, developers can control the appearance of various elements on a webpage, including colors, fonts, spacing, and positioning, thus enhancing the visual appeal and user experience of websites.

CSS offers three main methods for styling web pages: inline, internal, and external stylesheets.

1. \*\*Inline Styles:\*\* Inline styles are applied directly within HTML elements using the `style` attribute. This method allows for styling specific elements individually, but it's less efficient for larger projects as it mixes content with presentation. For example:

```html

<p style="color: blue; font-size: 16px;">This is a paragraph with inline styles.</p>

```

2. \*\*Internal Styles:\*\* Internal styles are defined within the `<style>` element in the HTML `<head>` section. This method allows for styling multiple elements within a single document, providing better organization than inline styles. For example:

```html

<head>

<style>

p {

color: blue;

font-size: 16px;

}

</style>

</head>

```

3. \*\*External Stylesheets:\*\* External stylesheets are separate CSS files linked to HTML documents using the `<link>` element. This method promotes modularity and reusability by allowing the same stylesheet to be applied across multiple pages. For example:

```html

<head>

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

</head>

```

And in `styles.css`:

```css

p {

color: blue;

font-size: 16px;

}

```

In summary, CSS provides versatile methods for styling web pages, allowing developers to create visually appealing and consistent designs while maintaining code organization and efficiency. Whether using inline, internal, or external stylesheets, CSS empowers developers to customize the look and feel of their websites to suit their design goals and user preferences.

**Expt 3 - Responsive Website with Bootstrap**

\*\*Theoretical Background:\*\*

Responsive web design has become a cornerstone in modern web development, ensuring that websites adapt seamlessly to various screen sizes and devices. By utilizing HTML and CSS, developers can dynamically resize, hide, or rearrange elements to maintain optimal user experience across desktops, tablets, and smartphones. This approach not only enhances accessibility but also addresses the evolving landscape of device usage among internet users.

Bootstrap, a popular open-source front-end framework, simplifies the process of creating responsive and mobile-first websites. It provides a comprehensive collection of HTML, CSS, and JavaScript components and tools, empowering developers to build consistent and visually appealing web pages efficiently. Bootstrap's key features, such as platform independence, responsive design capabilities, and customizable themes, contribute to its widespread adoption in the web development community.

Applications of Bootstrap extend beyond responsive web design, encompassing mobile-first development, cross-browser compatibility, and time-saving efficiency. Developers can integrate Bootstrap into their projects through two primary methods: including Bootstrap via Content Delivery Network (CDN) links or downloading the framework from the official website and linking to the CSS and JavaScript files locally.

\*\*Example:\*\*

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Responsive Web Design Example</title>

<!-- Bootstrap CSS library -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

</head>

<body>

<div class="container text-center">

<!-- Text color class used -->

<h1 class="text-success">Computer</h1>

<p>A computer is a machine that can be programmed to carry out sequences of arithmetic or logical operations (computation) automatically</p>

</div>

</body>

</html>

```

In this example, Bootstrap's CSS library is included via CDN links, enabling the use of Bootstrap classes to style elements and create a responsive layout.

**Expt 4 - Javascript Array, Objects and Functions**

\*\*Theoretical Background: JavaScript\*\*

JavaScript is a dynamic programming language primarily used for adding interactivity and dynamic behavior to web pages. Here are some key concepts and examples:

1. \*\*Variables and Data Types\*\*: Variables store data values, and JavaScript supports various data types such as strings, numbers, booleans, arrays, and objects.

```javascript

let name = "John";

let age = 25;

let isStudent = true;

let fruits = ["apple", "banana", "orange"];

let person = { name: "Alice", age: 30 };

```

2. \*\*Functions\*\*: Functions are blocks of code that perform a specific task. They can take parameters, execute operations, and return a result.

```javascript

function greet(name) {

return "Hello, " + name + "!";

}

let greeting = greet("World"); // Returns: "Hello, World!"

```

3. \*\*Arrays and Objects\*\*: Arrays store multiple values, and objects represent complex entities with key-value pairs.

```javascript

let colors = ["red", "green", "blue"];

console.log(colors[0]); // Output: "red"

let person = { name: "Alice", age: 30 };

console.log(person.name); // Output: "Alice"

```

4. \*\*Conditional Statements and Loops\*\*: Conditional statements execute code based on different conditions, and loops repeat code blocks multiple times.

```javascript

let age = 18;

if (age >= 18) {

console.log("You are an adult.");

} else {

console.log("You are a minor.");

}

for (let i = 0; i < 5; i++) {

console.log(i); // Output: 0, 1, 2, 3, 4

}

```

5. \*\*Sorting with Bubble Sort\*\*: Bubble Sort is a simple sorting algorithm used to sort arrays by repeatedly swapping adjacent elements if they are in the wrong order.

```javascript

function bubbleSort(arr) {

// Implementation of Bubble Sort

}

```

6. \*\*Average Grade Calculation\*\*: You can calculate the average grade of a class by summing up all grades and dividing by the total number of students.

```javascript

function calculateAverage(grades) {

// Implementation of average grade calculation

}

```

JavaScript's flexibility and simplicity make it a versatile language for building interactive web applications and enhancing user experiences.

**Expt 5 - Form Validation Using Javascript**

JavaScript is a powerful scripting language commonly used for client-side web development. One common application of JavaScript is form validation, where JavaScript code is used to ensure that user input meets certain criteria before it is submitted to the server.

In the provided example, a registration form is validated using JavaScript. The validateForm() function is called when the form is submitted. This function retrieves the values entered by the user in each form field and performs various checks to ensure that the input is valid.

For example, the function checks whether the user ID and password meet certain length requirements using simple conditional statements. It also checks whether the name contains only alphabetic characters using regular expressions. Additionally, it verifies whether the user has selected a country, ensures that the ZIP code consists of numeric characters only, and validates the email address format.

If any of the validation checks fail, an alert message is displayed to the user, informing them of the specific validation error. For instance, if the user ID is too short or too long, they will be alerted to adjust it accordingly.

This approach to form validation enhances user experience by providing immediate feedback on input errors, reducing the likelihood of submitting incorrect or incomplete data. By incorporating JavaScript validation, web developers can create more user-friendly and robust web forms, improving overall usability and data integrity.

**Expt 6 - DOM and DOM tree manipulation**

The Document Object Model (DOM) serves as a structured representation of HTML documents, enabling dynamic access and manipulation of their content, structure, and style. In the context of a to-do list application, DOM manipulation facilitates the creation, editing, deletion, and management of tasks in an interactive and user-friendly manner.

At the core of DOM manipulation lies the ability to select elements from the HTML document using JavaScript, allowing developers to interact with these elements programmatically. The `document.querySelector()` and `document.querySelectorAll()` methods are commonly used to retrieve elements based on CSS selectors, enabling precise targeting of specific elements within the document.

To create a to-do list application, event listeners are employed to respond to user interactions such as clicking buttons or pressing keys. When the user adds a new task, JavaScript dynamically creates DOM elements representing the task and appends them to the task list. Similarly, when a task is marked as completed or deleted, corresponding DOM elements are modified or removed accordingly.

Implementing editing functionality involves dynamically replacing task content with input fields when the user chooses to edit a task. Upon submitting the changes, the updated task content is reflected in the DOM, providing a seamless editing experience.

DOM manipulation allows developers to update the visual representation of tasks in real-time without requiring a page refresh, enhancing the application's responsiveness and usability. Moreover, by leveraging DOM events and methods, developers can implement interactive features that empower users to efficiently manage their tasks within the application.

In summary, DOM manipulation forms the backbone of a to-do list application, enabling developers to create a dynamic and intuitive user interface for task management. Through the selective targeting and modification of DOM elements, developers can seamlessly integrate user interactions and update the application's state in response to user actions.

**Expt 7 - JQuery**

jQuery is a popular JavaScript library designed to simplify DOM manipulation and event handling. It provides a concise and powerful set of methods for traversing and manipulating the DOM, making it easier for developers to create dynamic and interactive web applications.

For the first task, changing the color of a button using jQuery involves binding event handlers to the button element. jQuery's `.hover()` method can be used to specify two functions: one to be executed when the mouse enters the button (mouseover) and another to be executed when the mouse leaves the button (mouseout). Within these functions, jQuery's `.css()` method can be employed to dynamically modify the button's foreground and background colors, achieving the desired color change effect.

The second task focuses on inserting elements into the DOM using jQuery methods such as `.before()`, `.after()`, `.prepend()`, and `.append()`. These methods allow developers to add new elements before, after, at the beginning, or at the end of selected elements, respectively. This capability is particularly useful for dynamically updating the content of a webpage in response to user actions or other events.

For instance, `.before()` and `.after()` can be used to insert content adjacent to existing elements, while `.prepend()` and `.append()` can add content inside existing elements at the beginning and end, respectively. These methods enhance the flexibility and interactivity of web applications, enabling developers to seamlessly integrate new content or elements into the DOM structure.

Overall, jQuery's intuitive syntax and comprehensive set of methods empower developers to achieve complex DOM manipulation tasks with ease, ultimately enhancing the user experience and interactivity of web applications.

**Expt 8 - AJAX with HTML, XML and JSON data**

AJAX (Asynchronous JavaScript and XML) is a web development technique used to create dynamic and interactive web applications by making asynchronous requests to the server without reloading the entire webpage. It enables the retrieval and manipulation of data from a server asynchronously, allowing for seamless user experiences.

In the context of retrieving book information with AJAX, developers can utilize different data formats such as HTML, XML, and JSON to structure and transmit the data from the server to the client-side JavaScript code.

HTML (Hypertext Markup Language) is the standard markup language for creating web pages. It provides a simple and straightforward way to structure content on the web. With AJAX, HTML data can be retrieved asynchronously from the server and dynamically injected into the DOM (Document Object Model) to update the webpage's content without reloading.

XML (eXtensible Markup Language) is another data format commonly used for transmitting structured data over the web. XML is more flexible than HTML and allows developers to define custom markup languages for specific applications. AJAX requests can fetch XML data from the server, and JavaScript can parse and extract the relevant information for display on the webpage.

JSON (JavaScript Object Notation) has become increasingly popular for transmitting data between a server and a web application due to its lightweight and easy-to-parse format. AJAX requests can retrieve JSON data, which can be directly consumed by JavaScript as native objects, simplifying data manipulation and rendering.

By leveraging AJAX with HTML, XML, and JSON data formats, developers can create dynamic and interactive web applications that efficiently retrieve, process, and display information from a server, enhancing the overall user experience.

**Expt 9 - Inventory**

Managing an inventory system efficiently is crucial for businesses to track and control their stock levels. Node.js, combined with Express.js as the web framework and MongoDB as the database, provides a powerful and scalable solution for building such inventory management applications.

In this context, Node.js serves as the runtime environment, offering asynchronous and event-driven capabilities that enable high-performance handling of concurrent requests. Express.js complements Node.js by providing a robust and minimalist web framework, simplifying the process of building RESTful APIs for CRUD operations.

MongoDB, a NoSQL database, offers flexibility and scalability, making it an ideal choice for storing inventory data. Its document-oriented nature allows for the storage of inventory items with varying fields, such as name and quantity, in a schema-less format.

For creating new inventory items, the application utilizes the HTTP POST method to receive data from clients and store it in the MongoDB database. Retrieving a list of all inventory items involves using the HTTP GET method to fetch data from the database and send it back to clients.

Updating existing inventory items involves sending an HTTP PUT request with the updated information, which the application then processes to modify the corresponding document in the MongoDB collection.

Finally, deleting an inventory item entails sending an HTTP DELETE request with the item's unique identifier, enabling the application to locate and remove the corresponding document from the database.

Overall, by leveraging Node.js, Express.js, and MongoDB, developers can build a robust inventory management system that efficiently handles CRUD operations while offering scalability and flexibility to meet the evolving needs of businesses.