**DESCRIPTION::**

TASK---1

**LIST:**

An **HTML List**allows you to organize data on web pages into an ordered or unordered format to make the information easier to read and visually appealing. HTML Lists are very helpful for creating structured, accessible content in web development.

**Types of HTML Lists**

There are three main types of lists in HTML:

1. **Unordered Lists (<ul>):**These lists are used for items that do not need to be in any specific order. The list items are typically marked with bullets.
2. **Ordered Lists (<ol>):** These lists are used when the order of the items is important. Each item in an ordered list is typically marked with numbers or letters.
3. **Description Lists (<dl>):** These lists are used to contain terms and their corresponding descriptions.

**IMAGE:**

The HTML <img> tag is used to embed images in a web page. It is an empty or self-closing tag, meaning it doesn’t have a closing tag. It allows to display images from various sources, such as files on a website or URLs from other websites.

**LINK:**

The **HTML <link> tag**defines the relationship between the current document and an external resource, often for stylesheets or favicons. It’s an empty element with attributes like href and rel.

**TASK----2**

TABLE:

**HTML table tag** is used to display data in tabular form (row \* column). There can be many columns in a row.

We can create a table to display data in tabular form, using <table> element, with the help of <tr> , <td>, and <th> elements.

In Each table, table row is defined by <tr> tag, table header is defined by <th>, and table data is defined by <td> tags.

HTML tables are used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc. But it is recommended to use div tag over table to manage the layout of the page.

**FORM:**

The HTML <form> element provide a document section to take input from user. It provides various interactive controls for submitting information to web server such as text field, text area, password field, etc.

An **HTML form** is *a section of a document* which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

An HTML form facilitates the user to enter data that is to be sent to the server for processing such as name, email address, password, phone number, etc. .

**FRAME:**

HTML <frame> tag is used to divide web browser windows into multiple sections, each capable of loading content independently. This is achieved using a collection of frames within a frameset tag.

The **HTML <frameset> tag** was used in older versions of HTML to create multi-pane layouts within a web page. By dividing the browser window into multiple sections, each frame could display a separate HTML document. This allowed for the creation of complex web layouts with different content in each pane.

**TASK----3**

HTML5 (HyperText Markup Language 5) is the latest version of HTML, designed to structure and present content on the web. It introduces new elements, attributes, and behaviors to create more dynamic and interactive web pages.

### ****Features of HTML5****

* **Semantic Elements**: <header>, <footer>, <article>, <section> for better content organization.
* **Multimedia Support**: <audio>, <video> tags for embedding media without external plugins.

CSS (Cascading Style Sheets) is used to style and layout of web pages, and controlling the appearance of HTML elements. CSS targets HTML elements and applies style rules to dictate their appearance.

Below are the types of CSS:

* Inline CSS
* Internal or Embedded CSS
* External CSS

HTML5 and CSS3 work together to build modern, responsive, and visually appealing web pages. Understanding different CSS types helps developers apply styles effectively while maintaining efficient web development practices.

**TASK----4**

CSS selectors allow developers to style specific HTML elements by targeting them based on their attributes, relationships, or positions in the document.

### ****1. Simple Selectors****

* **Element Selector (tagname)** – Selects elements by tag name.
* **ID Selector (#id)** – Selects an element by its unique ID.
* **Class Selector (.class)** – Selects elements by class.
* **Group Selector (selector1, selector2)** – Applies styles to multiple selectors.
* **Universal Selector (\*)** – Selects all elements.

### ****2. Combinator Selectors****

* **Descendant Selector (ancestor descendant)** – Selects all elements inside a specified element.
* **Child Selector (parent > child)** – Selects direct child elements.
* **Adjacent Sibling Selector (element1 + element2)** – Selects an element immediately following another.
* **General Sibling Selector (element1 ~ element2)** – Selects all siblings after a specified element.

### ****3. Pseudo-Class Selectors****

* Target elements in a specific state (e.g., :hover, :focus, :nth-child(n)).

### ****4. Pseudo-Element Selectors****

* Style specific parts of an element (e.g., ::first-letter, ::before, ::after).

### ****5. Attribute Selectors****

* Style elements based on attribute values (e.g., [type="text"], [href^="https"]).

**TASK---5**

CSS (Cascading Style Sheets) is a styling language used to describe the presentation of a web page written in HTML. It allows developers to apply colors, background styles, fonts, text effects, and layout models to enhance the appearance of web pages.

CSS color:

CSS provides various ways to apply colors to elements:

### ****Ways to Define Colors in CSS****

1. **Named Colors** – e.g., red, blue, green
2. **Hex Code** – e.g., #FF5733
3. **RGB Values** – e.g., rgb(255, 87, 51)
4. **RGBA Values** – e.g., rgba(255, 87, 51, 0.5) (adds transparency)
5. **HSL Values** – e.g., hsl(120, 100%, 50%)
6. **HSLA Values** – e.g., hsla(120, 100%, 50%, 0.5)

## ****CSS Background****

The CSS background property is used to set the background of an element.

### ****Background Properties****

1. **background-color** – Sets a background color.
2. **background-image** – Sets a background image.
3. **background-repeat** – Specifies how the background image is repeated.
4. **background-position** – Sets the position of the background image.
5. **background-size** – Specifies the size of the background image.
6. **background-attachment** – Specifies whether the background scrolls with the page.

## ****CSS Font****

The font properties define the text appearance, including typeface, size, weight, and style.

### ****Common Font Properties****

1. **font-family** – Defines the font type (e.g., Arial, Times New Roman, etc.).
2. **font-size** – Sets the text size.
3. **font-weight** – Defines the thickness of text (normal, bold, etc.).
4. **font-style** – Controls the text style (normal, italic, oblique).
5. **font-variant** – Changes the text to small-caps.

## ****CSS Text****

CSS provides various properties to control text appearance and formatting.

### ****Common Text Properties****

1. **color** – Changes the text color.
2. **text-align** – Aligns text (left, right, center, justify).
3. **text-decoration** – Adds effects like underline, overline, line-through.
4. **text-transform** – Converts text to uppercase, lowercase, capitalize.
5. **letter-spacing** – Adjusts space between characters.
6. **word-spacing** – Adjusts space between words.
7. **line-height** – Controls the spacing between lines.

## ****CSS Box Model****

The **CSS Box Model** is the foundation of layout design. Every HTML element is treated as a rectangular box consisting of the following:

1. **Content** – The actual text or image inside the element.
2. **Padding** – The space between the content and the border.
3. **Border** – The outline surrounding the padding.
4. **Margin** – The space outside the border that separates elements.

### ****Box Model Properties****

* width & height – Define the size of the content area.
* padding – Creates space inside the border.
* border – Adds a visible boundary around the element.
* margin – Creates space outside the border.

**CSS styling** including **color, background, fonts, text, and the box model**. Mastering these concepts will help you design visually appealing and well-structured web pages.

**TASK-----6**

### ****Internal JavaScript****

### Internal JavaScript is written within an HTML file using the <script> tag inside the <head> or <body> section.

### ****External JavaScript****

External JavaScript is stored in a separate .js file and linked to an HTML file using the <script> tag with a src attribute.

* Improves code reusability and readability.
* Enhances maintainability.
* Keeps HTML and JavaScript separate.

## ****JavaScript Input/Output (I/O) Operations****

### ****Output Methods****

1. **Using console.log()**  
   Displays output in the browser console (useful for debugging).
2. **Using document.write()**  
   Writes output directly to the HTML document.
3. **Using alert()**  
   Displays a pop-up message.
4. **Using innerHTML**  
   Inserts output inside an HTML element.

### ****Input Methods****

1. **Using prompt()**  
   Takes user input through a pop-up box.
2. **Using confirm()**  
   Displays a confirmation dialog with "OK" and "Cancel".

## ****JavaScript Type Conversion****

### ****Implicit Type Conversion (Type Coercion)****

JavaScript automatically converts one data type into another when required.

### ****Explicit Type Conversion****

Using JavaScript functions to manually convert data types.

1. **String Conversion**
2. Number Conversion
3. Boolean Conversion
4. Using parseInt() and parseFloat()

* Internal and external JavaScript have their use cases; external scripts improve maintainability.
* JavaScript provides various I/O operations such as console.log(), alert(), prompt(), and DOM manipulation.
* Type conversion in JavaScript can be **implicit** (automatic) or **explicit** (using functions like String(), Number(), Boolean()).

**TASK-----7**

## ****Pre-defined Objects in JavaScript****

JavaScript provides several built-in objects that are ready to use. These objects simplify programming by offering pre-implemented methods and properties.

### ****Common Pre-defined Objects****

1. **Math Object** – Provides mathematical functions.
2. **Date Object** – Handles date and time.
3. **String Object** – Manages and manipulates text.
4. **Array Object** – Stores multiple values.
5. **RegExp Object** – Works with regular expressions.

## ****User-defined Objects in JavaScript****

JavaScript allows users to define custom objects to represent real-world entities.

### ****Ways to Create User-defined Objects****

1. **Using Object Literals**
2. **Using Constructor Functions**
3. **Using ES6 Classes**

* Pre-defined objects like Math, Date, String, and Array provide built-in functionality.
* User-defined objects help in structuring code in an object-oriented way.
* Constructor functions and ES6 classes allow better modularity and reusability.

**TASK---8**

### ****1. Conditional Statements****

Conditional statements are used to execute code based on certain conditions.

#### ****a. if Statement****

Executes a block of code only if the given condition evaluates to true.

#### ****b. if...else Statement****

Provides an alternative block of code when the condition evaluates to false.

#### ****c. if...else if...else Statement****

Used when multiple conditions need to be checked sequentially.

#### ****d. switch Statement****

Used when a variable is compared against multiple values.

### ****2.Loops in JavaScript****

Loops allow us to execute a block of code multiple times based on a condition.

#### ****a. for Loop****

A loop that runs a specified number of times.

#### ****b. while Loop****

Executes a block of code as long as the condition is true.

#### ****c. do...while Loop****

Similar to while, but guarantees execution of the code block at least once.

#### ****d. for...in Loop****

Used to iterate over the properties of an object.

#### ****e. for...of Loop****

Used to iterate over iterable objects like arrays and strings.

**TASK-----9**

## ****JavaScript Functions****

A **function** in JavaScript is a reusable piece of code that executes when called. Functions help make programs more modular and maintainable.

### ****1.1 Defining a Function****

A function can be defined using the function keyword.

### ****1.2 Calling a Function****

A function is executed when it is called (or invoked).

### ****1.3 Function with Parameters and Return Value****

Functions can take parameters (inputs) and return values.

### ****1.4 Anonymous Functions****

Functions without a name are called **anonymous functions**. They are often assigned to variables.

### ****1.5 Arrow Functions (ES6)****

Arrow functions provide a shorter syntax for defining functions.

## ****2. JavaScript Events****

Events allow JavaScript to interact with HTML elements. An event occurs when a user interacts with a webpage, such as clicking a button, pressing a key, or submitting a form.

### ****2.1 Common JavaScript Events****

| **Event** | **Description** |
| --- | --- |
| onclick | Triggered when an element is clicked. |
| onmouseover | Triggered when the mouse is over an element. |
| onmouseout | Triggered when the mouse leaves an element. |
| onkeydown | Triggered when a key is pressed. |
| onload | Triggered when a page finishes loading. |

### ****2.2 Adding Events Using HTML Attributes****

Events can be added directly inside an HTML element.

### ****2.3 Adding Events Using JavaScript****

Instead of using HTML attributes, events can be assigned using JavaScript.

### ****2.4 Event Listeners****

The addEventListener() method is a modern way to handle events.

### ****2.5 Removing Event Listeners****

An event listener can be removed using removeEventListener().

 **Functions** allow code reuse and modularity in JavaScript.

 **Events** enable interaction with HTML elements.

 Functions can accept **parameters** and return **values**.

 JavaScript events can be added using **HTML attributes**, **JavaScript event properties**, or **event listeners**.

 The addEventListener() method is preferred for handling events as it allows multiple event handlers for a single element.