# **Unit-1**

# **Explain any five tags used in head section in HTML5**

Certainly, here are five common HTML5 tags used in the **<head>** section:

1. **<meta>** tag:
   * Used for specifying metadata about the HTML document.
   * Attributes like **charset**, **name**, and **content** are commonly used for character encoding, keywords, and descriptions.
2. **<title>** tag:
   * Defines the title of the HTML document, which appears in the browser's title bar or tab.
   * Crucial for search engine optimization (SEO) and user experience.
3. **<link>** tag:
   * Typically used to include external resources, such as stylesheets (CSS) and icons.
   * Attributes like **rel**, **type**, and **href** are used to specify the relationship and location of the linked resource.
4. **<style>** tag:
   * Used for adding internal CSS styles directly within the HTML document.
   * Useful for creating unique styles for the document.
5. **<script>** tag:
   * Includes JavaScript code within the HTML document, either inline or by referencing external script files.
   * Important for adding interactivity and functionality to web pages.

These tags play a crucial role in structuring and enhancing the presentation and functionality of an HTML document.

# **List the global attributes of HTML5 Explain any four.**

Global attributes in HTML5 can be used with nearly any HTML element. Here are four global attributes along with explanations for each:

1. **class**:
   * The **class** attribute is used to specify one or more CSS classes that can be applied to an HTML element.
   * It allows you to define styles and apply them to multiple elements, promoting consistency in design.
2. **id**:
   * The **id** attribute provides a unique identifier for an HTML element within a web page.
   * It's used for targeting specific elements with JavaScript or CSS and is essential for creating anchors (links) within the same page.
3. **style**:
   * The **style** attribute allows you to provide inline CSS styles for an individual element.
   * This is useful when you want to apply specific styles directly to an element without the need for an external stylesheet.
4. **data-\*** (custom data attributes):
   * Custom data attributes allow you to store extra information with an element, which can be useful for scripting and data processing.
   * The attribute name starts with "data-" and can be any name you choose. For example, **data-info="some data"** can be accessed using JavaScript.

Global attributes like these are versatile and provide a way to manipulate and enhance the behavior and appearance of HTML elements across your web pages.

# **What is image map? Explain client side image map with example.**

**Image Map** is a feature in HTML that allows you to define multiple clickable areas (hotspots) within a single image. These hotspots can link to different web pages or perform various actions when clicked. There are two types of image maps: client-side and server-side. In this response, I'll explain client-side image maps and provide an example.

**Client-Side Image Map**:

* In a client-side image map, all the mapping information and behavior are defined within the HTML document itself, and interactivity is handled by the user's web browser without the need for a server.

**Example of a Client-Side Image Map**:

<!DOCTYPE html>

<html>

<head>

<title>Client-Side Image Map Example</title>

</head>

<body>

<h1>Click the regions on the map</h1>

<img src="worldmap.jpg" usemap="#worldmap" alt="World Map">

<map name="worldmap">

<!-- Define hotspots using <area> tags -->

<area shape="rect" coords="0,0,100,100" href="https://www.northamerica.com" alt="North America">

<area shape="circle" coords="200,150,50" href="https://www.europe.com" alt="Europe">

<area shape="polygon" coords="300,300,400,400,350,450" href="https://www.asia.com" alt="Asia">

</map>

</body>

</html>

In this example:

* We have an image (**<img>**) of a world map with a **usemap** attribute that references the map called "worldmap."
* The **<map>** element defines the image map named "worldmap."
* Inside the **<map>**, we use **<area>** tags to define three clickable regions (hotspots) on the image.
  + The **shape** attribute specifies the shape of the hotspot (rectangular, circular, or polygonal).
  + The **coords** attribute provides coordinates for the hotspot's shape.
  + The **href** attribute specifies the URL to navigate to when the hotspot is clicked.
  + The **alt** attribute provides alternative text for accessibility.

When a user clicks on any of these hotspots, they will be taken to the specified web page. This is a simple example of a client-side image map, which enhances user interactivity and navigation on web pages containing images with multiple clickable areas.

# **Give the complete HTML code for webpage as given below:**

# 

# **Differentiate between- TextArea& Textfield, Radiobutton & Checkbox.**

**Differentiation between TextArea and TextField:**

1. **TextArea:**
   * TextArea is a multi-line text input field in HTML.
   * It allows users to enter and edit multiple lines of text, making it suitable for longer paragraphs or user comments.
   * Defined using the **<textarea>** element.
   * Example: **<textarea rows="4" cols="50">This is a TextArea.</textarea>**
2. **TextField:**
   * TextField is a single-line text input field in HTML.
   * It is typically used for short, single-line inputs such as names, email addresses, or search queries.
   * Defined using the **<input>** element with the **type** attribute set to "text."
   * Example: **<input type="text" name="username" value="JohnDoe">**

**Differentiation between RadioButton and Checkbox:**

1. **RadioButton:**
   * RadioButtons are used when you want the user to select one option from a group of mutually exclusive options.
   * Only one RadioButton in a group can be selected at a time.
   * Defined using the **<input>** element with the **type** attribute set to "radio."
   * Example:

<input type="radio" name="gender" value="male"> Male

<input type="radio" name="gender" value="female"> Female

1. **Checkbox:**
   * Checkboxes are used when you want the user to select one or more options from a group of choices.
   * Users can select multiple checkboxes simultaneously.
   * Defined using the **<input>** element with the **type** attribute set to "checkbox."
   * Example:

<input type="checkbox" name="interest" value="music"> Music

<input type="checkbox" name="interest" value="sports"> Sports

<input type="checkbox" name="interest" value="reading"> Reading

In summary, TextArea is for multi-line text input, TextField is for single-line text input, RadioButton is for selecting one option from a group, and Checkbox is for selecting one or more options from a group. Each element serves a specific purpose in capturing user input.

# **Explain with any example properties used to set the font style of the text in HTML5**

In HTML5, you can set the font style of text using CSS (Cascading Style Sheets) properties. There are several CSS properties that allow you to control the font style of text. Here are some common font style properties along with an example:

1. **font-family**:
   * This property is used to specify the font family or typeface for the text.
   * It allows you to set the font to be used, such as Arial, Times New Roman, or a custom web font.

p { font-family: "Arial", sans-serif; }

1. **font-size**:
   * **font-size** determines the size of the text.
   * You can use various units like pixels (px), ems (em), percentages (%), or keywords (e.g., "small," "large") to set the font size.

h1 { font-size: 24px; }

1. **font-weight**:
   * This property controls the thickness or boldness of the text.
   * It can take values like "normal," "bold," or numeric values (e.g., 400, 700) for fine-tuning the weight.

strong { font-weight: bold; }

1. **font-style**:
   * **font-style** sets the style of the text, allowing you to use values like "normal," "italic," or "oblique" to slant the text.

em { font-style: italic; }

1. **text-decoration**:
   * This property controls text decorations like underlining, overlining, and striking through text.

a { text-decoration: underline; }

1. **color**:
   * **color** sets the color of the text. You can use color names, hexadecimal codes, RGB values, or HSL values.

p { color: #336699; }

Here's a complete example of using these properties to style text in HTML:

<!DOCTYPE html>

<html>

<head>

<style>

p {

font-family: "Arial", sans-serif;

font-size: 16px;

font-weight: bold;

font-style: italic;

text-decoration: underline;

color: #FF0000;

}

</style>

</head>

<body>

<p>This is a styled text example.</p>

</body>

</html>

This HTML document defines a **<p>** element with various font style properties applied to it, resulting in a styled text.

# **Explain any five text formatting tags in HTML5.**

HTML5 provides various text formatting tags that allow you to control the appearance and layout of text on web pages. Here are five text formatting tags in HTML5:

1. **<strong>** and **<b>**:
   * **<strong>** is a semantic tag used to represent important or strongly emphasized text, often displayed in bold by browsers. It indicates the text's significance to the content.
   * **<b>** is a presentational tag that historically represented bold text, but it's better to use **<strong>** for semantic meaning.

Example:

<p>This is <strong>important</strong> text.</p>

1. **<em>** and **<i>**:
   * **<em>** is a semantic tag used to emphasize text, typically displayed in italics. It indicates a level of emphasis or importance to the text.
   * **<i>** is a presentational tag that historically represented italic text, but it's better to use **<em>** for semantic meaning.

Example:

<p><em>This text is emphasized</em>, and <i>so is this</i>.</p>

1. **<u>**:
   * The **<u>** tag is used to underline text, although it's considered somewhat outdated in modern web design due to accessibility concerns and the use of CSS for styling.

Example:

<p>This is <u>underlined</u> text.</p>

1. **<sub>** and **<sup>**:
   * **<sub>** is used for subscript text, which appears lower than the normal text and is commonly used for chemical formulas and mathematical notations.
   * **<sup>** is used for superscript text, which appears higher than the normal text and is often used for exponents, footnotes, or trademark symbols.

Example:

<p>H<sub>2</sub>O is water, and E=mc<sup>2</sup> is a famous equation.</p>

1. **<del>** and **<ins>**:
   * **<del>** represents deleted or removed text, and browsers typically render it with a strikethrough line.
   * **<ins>** represents inserted or added text, and browsers may underline it or highlight it in some way.

Example:

<p><del>This is removed</del> and <ins>this is added</ins> text.</p>

These text formatting tags provide a way to add meaning and style to your web content, and it's a good practice to use semantic tags like **<strong>**, **<em>**, **<sub>**, and **<sup**> for better accessibility and search engine optimization.

# **What is Stylesheet? List its types. Explain any ONE with an example**

A **stylesheet** is a document or a set of rules used to define the visual presentation and layout of elements on a web page. Stylesheets are primarily used to control the appearance of HTML or XML documents. They allow web developers to apply styles like colors, fonts, margins, and positioning to elements, making it easier to create visually appealing and consistent web pages.

There are three main types of stylesheets:

1. **CSS (Cascading Style Sheets)**:
   * CSS is the most common and widely used type of stylesheet. It defines styles for HTML elements and controls how elements are displayed in web browsers. CSS can be applied internally (within an HTML document), externally (in a separate .css file), or inline (directly within HTML elements).

**Example of an External CSS Stylesheet:**

Suppose you have an HTML document named **index.html** and a separate CSS file named **styles.css**. Here's an example of an external CSS stylesheet:

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

**styles.css**:

/\* External CSS Stylesheet \*/

h1 {

color: #336699;

font-family: Arial, sans-serif;

}

p {

font-size: 16px;

margin-left: 20px;

}

In this example:

* The HTML file links to the external CSS file using the **<link>** element in the **<head>** section.
* The CSS file defines styles for the **<h1>** and **<p>** elements, setting their text color, font, font size, and margin.

When you open **index.html** in a web browser, the styles defined in **styles.css** are applied to the HTML elements, affecting their appearance.

The use of external CSS allows for centralized and consistent styling across multiple web pages, making it easier to maintain and update the visual design of a website.

# **Explain the following tags in HTML5 with an example-<img>, <link>, <div>, <span> , <title>**

Certainly, I'll explain each of the HTML5 tags you mentioned along with an example for each.

1. **<img> Tag**:
   * The **<img>** tag is used to embed images in an HTML document.
   * It doesn't have a closing tag and uses the **src** attribute to specify the image file's source.

Example:

<img src="example.jpg" alt="An example image">

1. **<link> Tag**:
   * The **<link>** tag is primarily used to link external resources such as stylesheets (CSS) or icon files to an HTML document.
   * It's placed within the **<head>** section.

Example (linking to an external CSS file):

<link rel="stylesheet" type="text/css" href="styles.css">

1. **<div> Tag**:
   * The **<div>** tag is a block-level container used to group and structure content on a web page.
   * It's often styled using CSS and doesn't have any default visual appearance.

Example:

<div>

<h2>This is a content container</h2>

<p>Content goes here...</p>

</div>

1. **<span> Tag:**
   * The **<span>** tag is an inline-level container used for styling and formatting a specific portion of text within a block-level element.
   * Like the **<div>** tag, it doesn't have default visual appearance.

Example:

<p>This is a <span style="color: blue;">blue</span> word.</p>

1. **<title> Tag**:
   * The **<title>** tag is used within the **<head>** section of an HTML document to specify the title of the webpage.
   * The title is displayed in the browser's title bar or tab and is essential for SEO.

Example:

<!DOCTYPE html>

<html>

<head>

<title>Example Web Page</title>

</head>

<body>

<h1>Welcome to the Example Web Page</h1>

<p>This is the content of the page.</p>

</body>

</html>

In this example, the **<title>** tag sets the title of the webpage to "Example Web Page." This is what the user would see in their browser's title bar or tab when they visit the page.

# **Which are the CSS properties to work with background of a Page in HIMI? Give its purpose with an example.**

It seems like you're referring to "HIMI," which might be a typo. I assume you mean HTML and CSS. In HTML and CSS, you can use various CSS properties to work with the background of a webpage. Here are some commonly used background-related CSS properties along with their purposes and examples:

1. **background-color**:
   * **Purpose**: Sets the background color of an element.
   * **Example**:

body { background-color: #f0f0f0; }

1. **background-image**:
   * **Purpose**: Specifies an image to be used as the background of an element.
   * **Example**:

.header { background-image: url('header-bg.jpg'); }

1. **background-repeat**:
   * **Purpose**: Defines how the background image should repeat, e.g., repeat, no-repeat, repeat-x, or repeat-y.
   * **Example**:

.footer { background-image: url('footer-bg.png'); background-repeat: repeat-x; }

1. **background-position**:
   * **Purpose**: Sets the starting position of the background image within the element.
   * **Example**:

.sidebar { background-image: url('sidebar-bg.png'); background-position: top right; }

1. **background-size**:
   * **Purpose**: Specifies the size of the background image, either as a percentage, specific dimensions, or keywords like "cover" or "contain."
   * **Example**:

.jumbotron { background-image: url('jumbotron-bg.jpg'); background-size: cover; }

1. **background-attachment**:
   * **Purpose**: Determines whether the background image scrolls with the content or remains fixed as the page is scrolled.
   * **Example**:

.parallax { background-image: url('parallax-bg.jpg'); background-attachment: fixed; }

1. **background-blur** (Note: This is not a standard CSS property but can be achieved using CSS filters and backdrop-filter):
   * **Purpose**: Applies a blur effect to the background, creating a frosted glass or translucent effect.
   * **Example**:

.modal { background-image: url('modal-bg.jpg'); background-blur: 5px; }

These CSS properties allow you to customize the background of HTML elements, providing you with the flexibility to create visually appealing and dynamic webpage designs.

# **Write HTML code to generate a form accepting the user last name (textfield). Contact number(textfield), gender (radiobutton), hobbies (checkbox), country (drodown) and buttons for resetting and submitting the form**

Certainly! Here's an example of HTML code for a form that accepts the user's last name, contact number, gender (radio buttons), hobbies (checkboxes), country (dropdown), and buttons for resetting and submitting the form:

<!DOCTYPE html>

<html>

<head>

<title>User Information Form</title>

</head>

<body>

<h2>User Information Form</h2>

<form action="process\_form.php" method="post">

<!-- Last Name -->

<label for="lastname">Last Name:</label>

<input type="text" id="lastname" name="lastname" required><br><br>

<!-- Contact Number -->

<label for="contact">Contact Number:</label>

<input type="text" id="contact" name="contact" required><br><br>

<!-- Gender -->

<label>Gender:</label><br>

<input type="radio" id="male" name="gender" value="male">

<label for="male">Male</label><br>

<input type="radio" id="female" name="gender" value="female">

<label for="female">Female</label><br>

<input type="radio" id="other" name="gender" value="other">

<label for="other">Other</label><br><br>

<!-- Hobbies -->

<label>Hobbies:</label><br>

<input type="checkbox" id="reading" name="hobbies[]" value="Reading">

<label for="reading">Reading</label><br>

<input type="checkbox" id="sports" name="hobbies[]" value="Sports">

<label for="sports">Sports</label><br>

<input type="checkbox" id="music" name="hobbies[]" value="Music">

<label for="music">Music</label><br><br>

<!-- Country -->

<label for="country">Country:</label>

<select id="country" name="country">

<option value="usa">United States</option>

<option value="canada">Canada</option>

<option value="uk">United Kingdom</option>

<option value="australia">Australia</option>

<option value="other">Other</option>

</select><br><br>

<!-- Buttons for Reset and Submit -->

<input type="reset" value="Reset">

<input type="submit" value="Submit">

</form>

</body>

</html>

In this code:

* The **form** element defines the user information form. It specifies the **action** attribute (the URL where the form data will be sent) and the **method** attribute (the HTTP method, which is POST in this case).
* Text fields for last name and contact number are created using **input** elements with the **type** attribute set to "text."
* Radio buttons for gender and checkboxes for hobbies are created using **input** elements with appropriate **type** values.
* The country dropdown is created using a **select** element with **option** elements specifying the available countries.
* Reset and Submit buttons are created using **input** elements with **type** attributes set to "reset" and "submit."

You can replace "process\_form.php" in the form's **action** attribute with the URL where you want to process the form data. Additionally, you can customize the form elements and options as needed.

# **Write an HTML and CSS code for the following**

# 

# **Explain any five text formatting tags in HTML5.**

HTML5 provides a variety of text formatting tags and attributes that allow you to control the presentation and styling of text on a web page. Here are five text formatting tags in HTML5:

1. **<strong> and <b>**:
   * **<strong>**: Specifies that the text within it is of strong importance, typically rendered as bold.
   * **<b>**: Represents text that should be stylistically bold without conveying additional semantic importance.
   * **Example**:

<p>This is <strong>important</strong> text, and this is <b>bold</b> text.</p>

1. **<em> and <i>**:
   * **<em>**: Indicates emphasized text, usually rendered as italic.
   * **<i>**: Represents text in an alternate voice or mood, often rendered as italic.
   * **Example**:

<p>This is <em>emphasized</em> text, and this is <i>italicized</i> text.</p>

1. **<u>**:
   * Specifies underlined text. However, it's important to note that using underlines for non-hyperlinked text is discouraged due to potential confusion with hyperlinks.
   * **Example**:

<p>This is <u>underlined</u> text.</p>

1. **<sup> and <sub>**:
   * **<sup>**: Represents superscript text, which is typically used for mathematical notations, exponents, or footnotes.
   * **<sub>**: Represents subscript text, often used for chemical formulas or mathematical subscripts.
   * **Example**:

<p>The chemical formula for water is H<sub>2</sub>O, and 2<sup>3</sup> equals 8.</p>

1. **<del> and <ins>**:
   * **<del>**: Represents text that has been deleted or removed from the document.
   * **<ins>**: Indicates text that has been inserted or added to the document.
   * **Example**:

<p>He said <del>goodbye</del> and then <ins>hello</ins> again.</p>

These text formatting tags help convey the intended meaning and presentation of text within your HTML documents. However, it's important to note that in modern web development, CSS is often preferred for controlling text styles, fonts, and formatting, while HTML tags are used more for structuring content and providing semantic meaning.

# **How to create links in HTML5? Explain its tags with an example.**

In HTML5, you can create links using the **<a>** (anchor) element. The **<a>** element is used to define hyperlinks, allowing users to navigate to other web pages or resources. Here's how to create links in HTML5 using the **<a>** element:

**Syntax**:

<a href="URL">Link Text</a>

* **href**: This attribute specifies the destination URL or resource to which the link points. It can be an absolute URL (e.g., "[https://www.example.com](https://www.example.com/)") or a relative URL (e.g., "page.html"). It can also point to email addresses or other resources like files.
* **Link Text**: This is the text that will be displayed as a clickable link.

**Examples**:

1. Creating a basic external link to a website:

<a href="https://www.example.com">Visit Example.com</a>

1. Creating a relative link to an internal page within the same website:

<a href="about.html">Learn More About Us</a>

1. Creating a link to send an email:

<a href="mailto:info@example.com">Send us an email</a>

1. Creating a link to download a file:

<a href="document.pdf" download>Download Document</a>

1. Creating a link with a target attribute to open in a new tab or window:

<a href="https://www.example.com" target="\_blank">Open in a new tab</a>

1. Creating a link with additional attributes for accessibility:

<a href="page.html" title="Read more about our products" aria-label="Product Page" tabindex="1">Products</a>

The **<a>** element is a fundamental building block for creating hyperlinks on web pages. When a user clicks on the link, their browser navigates to the specified URL, opening a new page or resource if indicated. You can use CSS to style the appearance of the link, and JavaScript to add interactivity if needed.

# **Define image map? Explain <img> tag with four attributes**

**Image Map**: An image map is an HTML feature that allows you to associate different links or actions with specific regions of an image. These regions are defined using various shapes (rectangles, circles, and polygons) and are typically used to create interactive and clickable areas within an image. Image maps are useful for creating navigation menus, clickable diagrams, or any other interactive graphics where different actions or links are associated with specific parts of the image.

**The <img> Tag and Four Common Attributes**:

The **<img>** tag is used in HTML to embed images in a web page. Here are four common attributes used with the **<img>** tag:

1. **src (Source)**:
   * The **src** attribute specifies the URL of the image file to be displayed. This is a required attribute.
   * Example:

<img src="image.jpg" alt="A beautiful landscape">

1. **alt (Alternative Text)**:
   * The **alt** attribute provides alternative text for the image, which is displayed if the image cannot be loaded or by assistive technologies for accessibility.
   * Example:

<img src="image.jpg" alt="A beautiful landscape">

1. **width and height**:
   * The **width** and **height** attributes specify the dimensions (in pixels) of the image. These attributes are optional, but it's a good practice to set them to the actual width and height of the image to prevent layout shifts as the image loads.
   * Example:

<img src="image.jpg" alt="A beautiful landscape" width="800" height="600">

1. **title**:
   * The **title** attribute provides additional information about the image when the user hovers the cursor over it. It is used for tooltips and can be beneficial for providing more context.
   * Example:

htmlCopy code

<img src="image.jpg" alt="A beautiful landscape" title="Scenic view of a lake surrounded by mountains">

These attributes are essential for specifying and enhancing the appearance and accessibility of images within your web pages. Additionally, the **<img>** tag can be combined with other HTML features, like image maps, to create interactive and engaging content.

# **Explain following CSS background properties with an example-bacground-repeat, background-position, background-color**

Certainly! CSS provides several background properties to control how background images are displayed on an element. Here, we'll explain the following CSS background properties with examples:

1. **background-repeat**:
   * The **background-repeat** property controls whether and how a background image is repeated both horizontally and vertically. It takes one of the following values:
     + **repeat**: The background image is repeated both horizontally and vertically (the default behavior).
     + **repeat-x**: The background image is repeated only horizontally.
     + **repeat-y**: The background image is repeated only vertically.
     + **no-repeat**: The background image is not repeated at all.
   * **Example**:

.container { background-image: url('pattern.png'); background-repeat: repeat; }

In this example, the **pattern.png** image will be repeated both horizontally and vertically as the background of the **.container** element.

1. **background-position**:
   * The **background-position** property specifies the starting position of the background image within the element. It can be defined using keywords, percentages, or specific coordinates.
   * **Example**:

.header { background-image: url('header-bg.jpg'); background-position: center top; }

This code sets the background image for the **.header** element and positions it in the center horizontally and at the top vertically.

1. **background-color**:
   * The **background-color** property sets the background color for an element, and it is often used without a background image. It takes a color value (e.g., a named color, hexadecimal value, or RGB value).
   * **Example**:

.callout { background-color: #f0f0f0; }

In this example, the background color of the **.callout** element is set to a light gray color.

These background properties offer control over how background images are repeated, positioned, and colored, allowing you to create visually appealing and customized designs for your web page elements.

# **Define Style sheet. What is the purpose of following CSS Text style properties in HTML5 text-align, text-transform, text-decoration, color**

**Style Sheet**: A style sheet is a document or file that contains a set of rules and declarations used to control the visual presentation of elements on a web page. Style sheets are essential in web development for defining the layout, formatting, and styling of HTML elements, including text, images, and other content. There are two main types of style sheets in web development: Cascading Style Sheets (CSS) and Inline Styles.

* **Cascading Style Sheets (CSS)**: These are external or internal files containing style rules that apply to multiple elements throughout a website. CSS allows for separation of content (HTML) and presentation (styling), making it easier to maintain and update the design of a site.
* **Inline Styles**: These are styles applied directly to individual HTML elements using the **style** attribute. Inline styles are typically used for specific, one-off style adjustments for a single element.

**CSS Text Style Properties**:

The CSS text style properties, such as **text-align**, **text-transform**, **text-decoration**, and **color**, are used to control the appearance and formatting of text within HTML elements. Here's a brief explanation of each property:

1. **text-align**:
   * The **text-align** property determines the horizontal alignment of text content within an element. It can take values like "left," "right," "center," or "justify."
   * **Purpose**: It helps control the positioning of text within its containing element, ensuring proper alignment.
2. **text-transform**:
   * The **text-transform** property changes the capitalization of text. It can be set to values like "uppercase," "lowercase," or "capitalize."
   * **Purpose**: It modifies the appearance of text by transforming it to uppercase, lowercase, or capitalizing the first letter of each word.
3. **text-decoration**:
   * The **text-decoration** property is used to add or remove text decorations such as underlines, overlines, line-throughs, and blink effects.
   * **Purpose**: It is used for enhancing or modifying the visual style of text, making it more visually engaging or indicating specific text states.
4. **color**:
   * The **color** property sets the color of the text content. It accepts color values in various formats (e.g., named colors, hexadecimal values, RGB values).
   * **Purpose**: It determines the color of the text, allowing you to control the visual appearance of text elements.

These CSS text style properties are essential for fine-tuning the appearance of text on a web page, ensuring readability, consistency, and visual appeal. They play a crucial role in making web content more accessible and engaging for users.

Top of Form

# **Give the complete form code for following in HTML5-**

# 

# **Unit-2**

# **Write a Java Script code to accept lower and upper range in number. Display sum of all even numbers between the ranges**

Here's a JavaScript code snippet that accepts lower and upper range values and calculates the sum of all even numbers between those ranges:

function sumOfEvenNumbers(lower, upper) {

if (lower > upper) {

return "Lower range should be less than or equal to upper range.";

}

let sum = 0;

for (let i = lower; i <= upper; i++) {

if (i % 2 === 0) {

sum += i;

}

}

return sum;

}

// Example usage:

const lowerRange = parseInt(prompt("Enter the lower range:"));

const upperRange = parseInt(prompt("Enter the upper range:"));

const result = sumOfEvenNumbers(lowerRange, upperRange);

if (typeof result === "number") {

console.log(`Sum of even numbers between ${lowerRange} and ${upperRange} is: ${result}`);

} else {

console.log(result);

}

This code defines a **sumOfEvenNumbers** function that takes the lower and upper range values as parameters. It then iterates through the numbers between the lower and upper ranges, checking if each number is even (i.e., divisible by 2). If it's even, the number is added to the **sum**. Finally, the function returns the sum of even numbers.

The example usage section prompts the user to enter the lower and upper range, calculates the sum, and displays the result.

# **What is locallon object? Explain its two methods with example.**

It appears that you may be referring to the "Location" object in JavaScript, which represents the current URL of the web page and provides methods and properties to work with it. Here are two commonly used methods of the **Location** object in JavaScript:

1. **location.href**:
   * The **location.href** property is used to get or set the entire URL of the current page, including the protocol, domain, path, query parameters, and the fragment identifier (e.g., "<https://www.example.com/index.html?query=example#section>").
   * You can use it to navigate to a different URL or to extract information from the current URL.

**Example**:

// Get the current URL

const currentURL = location.href;

console.log("Current URL:", currentURL);

// Change the URL and navigate to a new page

location.href = "https://www.example.com/new-page.html";

1. **location.reload()**:
   * The **location.reload()** method is used to reload the current page. It has an optional parameter that determines whether to force a reload from the server (true) or use the browser's cache (false).
   * This method is often used to refresh the page when needed, such as after making changes to the page's content.

**Example**:

// Reload the current page from the server (ignoring cache)

location.reload(true);

The **Location** object provides several other properties and methods for working with URLs and navigating between web pages. It's commonly used in web development to create dynamic and interactive web applications.

# **Discuss Document Object Model**

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of an HTML or XML document as a tree-like data structure, where each element in the document is represented as an object in the tree. The DOM allows programmers to interact with web pages using a programming language like JavaScript.

Key points about the DOM include:

1. **Tree Structure**: The DOM represents an HTML or XML document as a hierarchical tree structure. This structure is made up of nodes, and each node corresponds to a part of the document, such as elements, attributes, and text.
2. **Accessible and Manipulable**: The DOM provides a way for programs to access and manipulate the content and structure of a web page. You can change the content, structure, and style of the page in real-time using JavaScript.
3. **Platform and Language Independence**: The DOM is not tied to any specific programming language or platform. Although JavaScript is the most common language for DOM manipulation, other languages can also interact with the DOM, making it highly versatile.
4. **Dynamic and Interactive**: The DOM allows for dynamic and interactive web pages. Elements on a page can be created, modified, or removed on the fly, enabling user interactions and real-time updates.
5. **Consistency**: The DOM provides a consistent way to access and modify web documents across different browsers. This consistency is crucial for cross-browser compatibility.
6. **Standardized Interfaces**: The DOM is defined by the World Wide Web Consortium (W3C), and it provides standardized interfaces and methods for working with documents.

Here's a simple example of how the DOM can be accessed and manipulated using JavaScript:

<!DOCTYPE html>

<html>

<head>

<title>DOM Example</title>

</head>

<body>

<h1 id="heading">Hello, DOM!</h1>

<p>This is a paragraph.</p>

<script>

// Access the element with id "heading"

var heading = document.getElementById("heading");

// Change the text content and style

heading.innerHTML = "Modified heading";

heading.style.color = "blue";

</script>

</body>

</html>

In this example, JavaScript is used to access the **h1** element with the id "heading" and change its text content and style.

The DOM is a fundamental concept in web development, as it forms the basis for creating dynamic and interactive web pages and applications. It allows for a seamless interaction between the web page's content and the code running in the browser.

# **Explain the uses of Window object methods: Prompt, Alert**

The **Window** object in JavaScript represents the browser window or frame. It provides methods and properties to interact with the browser, including displaying dialog boxes and alerts. Two commonly used methods of the **Window** object are **prompt()** and **alert()**:

1. **alert() Method**:
   * The **alert()** method displays a dialog box with a specified message and an "OK" button. It is commonly used to provide information or notify the user of something.
   * It is a simple way to communicate with the user, but it doesn't allow user input.

**Usage**:

window.alert("This is an alert message.");

1. **prompt() Method**:
   * The **prompt()** method displays a dialog box that prompts the user to enter text or data. It allows for user input, and the entered data can be captured and used in your JavaScript code.
   * It is often used for getting user input in the form of text.

**Usage**:

var userInput = window.prompt("Please enter your name:");

if (userInput !== null) {

console.log("Hello, " + userInput + "!");

}

Here are some common use cases for these methods:

* **alert()**:
  + Displaying informative messages or notifications to the user.
  + Alerting the user of successful or failed actions.
  + Providing feedback or instructions.
  + Displaying a warning message when an error occurs.
  + Confirming an action when leaving a page or submitting a form (via an "onbeforeunload" event).
* **prompt()**:
  + Collecting user input, such as names, email addresses, or other textual data.
  + Creating interactive forms and dialogs.
  + Capturing data for use in JavaScript functions or calculations.
  + Implementing simple interactive features on a web page.

Both methods are simple but effective ways to interact with users and obtain input or provide information. However, they are often considered basic and can be enhanced with more sophisticated dialog and user interaction features through HTML and CSS, or by using more advanced JavaScript libraries and frameworks.

# **Describe the features of XML.**

Extensible Markup Language (XML) is a versatile and widely used markup language that is designed to store and transport data. XML is characterized by several key features that make it a valuable tool for structuring and sharing information. Here are some of the main features of XML:

1. **Self-Descriptive**:
   * XML documents are self-descriptive, meaning they contain data and metadata that describe the data's structure. This makes it easy to understand and interpret the content.
2. **Hierarchical Structure**:
   * XML documents are organized in a hierarchical structure, consisting of elements within elements, forming a tree-like structure. This hierarchy allows for the representation of complex data relationships.
3. **Human-Readable**:
   * XML is human-readable, with data stored in plain text, making it easy for humans to read and understand the content without the need for specialized tools.
4. **Platform-Independent**:
   * XML is platform-independent and can be used on different operating systems and with various programming languages. It promotes interoperability across different systems.
5. **Extensible**:
   * XML is extensible, which means that you can define your own custom elements and attributes to suit your specific data representation needs. This flexibility is one of the key features of XML.
6. **Structured Data**:
   * XML is suitable for representing structured data, such as configuration files, data interchange formats, and data storage. It provides a consistent way to organize data.
7. **Data Validation**:
   * XML documents can be validated against Document Type Definitions (DTDs) or XML Schemas to ensure data consistency and conformity to predefined rules.
8. **Data Transformation**:
   * XML can be transformed into different formats using technologies like XSLT (Extensible Stylesheet Language Transformations) to present data in a variety of ways, such as HTML or other XML formats.
9. **Data Interchange**:
   * XML is commonly used for data interchange between different systems, applications, and platforms. It serves as a neutral format that various applications can understand and work with.
10. **Standardized**:
    * XML is a well-established and widely adopted standard, supported by many programming languages and technologies. This standardization ensures its longevity and compatibility.
11. **Namespace Support**:
    * XML allows for the use of namespaces, which helps prevent naming conflicts when combining XML from different sources.
12. **Well-Defined Specification**:
    * The XML specification is well-defined by the World Wide Web Consortium (W3C), which helps ensure consistency and interoperability.
13. **Use Cases**:
    * XML is used in various applications, including data exchange in web services (SOAP), configuration files (e.g., for web servers), document storage, and data representation in various domains such as finance and healthcare.

XML's features make it a versatile tool for representing and transporting structured data, and it continues to be a significant technology for data exchange, configuration, and document storage in many domains.

# **Write DTD and XML code for describing invoice, each invoice includes: Invoice number, Date of purchase Billing address, Name of purchaser, List of terms giving code number, quantity ordered, price , Total amount of the order.**

To describe an XML structure for invoices with a Document Type Definition (DTD), you can create an XML structure that adheres to the defined DTD rules. Here's an example of an XML structure for invoices and its corresponding DTD:

**DTD (invoice.dtd)**:

<!ELEMENT invoices (invoice+)>

<!ELEMENT invoice (invoicenumber, dateofpurchase, billingaddress, purchaser, terms+, totalamount)>

<!ELEMENT invoicenumber (#PCDATA)>

<!ELEMENT dateofpurchase (#PCDATA)>

<!ELEMENT billingaddress (#PCDATA)>

<!ELEMENT purchaser (#PCDATA)>

<!ELEMENT terms (code, quantity, price)>

<!ELEMENT code (#PCDATA)>

<!ELEMENT quantity (#PCDATA)>

<!ELEMENT price (#PCDATA)>

<!ELEMENT totalamount (#PCDATA)>

This DTD defines a structure for invoices where each invoice includes an invoice number, date of purchase, billing address, name of the purchaser, a list of terms, and the total amount of the order.

**XML (invoice.xml)**:

<?xml version="1.0"?>

<!DOCTYPE invoices SYSTEM "invoice.dtd">

<invoices>

<invoice>

<invoicenumber>12345</invoicenumber>

<dateofpurchase>2023-10-01</dateofpurchase>

<billingaddress>123 Main St, City, Country</billingaddress>

<purchaser>John Doe</purchaser>

<terms>

<code>ABC123</code>

<quantity>5</quantity>

<price>10.00</price>

</terms>

<terms>

<code>XYZ789</code>

<quantity>3</quantity>

<price>20.00</price>

</terms>

<totalamount>115.00</totalamount>

</invoice>

<invoice>

<!-- Another invoice entry -->

</invoice>

</invoices>

In this example XML document, there are two invoice entries. Each invoice contains the requested information, including invoice number, date of purchase, billing address, purchaser name, terms (code, quantity, price), and the total amount of the order. The **<!DOCTYPE>** declaration references the DTD file, ensuring that the XML adheres to the defined structure.

You can extend this XML structure and DTD as needed to include more invoice entries or additional details in each invoice. The DTD defines the rules and structure, ensuring data consistency and validation.

# **Explain the following with valid example: alert(), prompt()**

# **What is the purpose of Location browser objects in JavaScript? Explain its three properties**

# **write a short note on XML DTD**

# **Explain the following elements w.r.t XSLT with an example: xsl: attribute, xsl:value-of, xsl:attribute-set**

# **Describe Document Object Model with diagram**

# **Write a JavaScript code to read a number from a text filed and check if it is Armstrong Number or not (Assume the input to be always three digit number)**

# **What is the use of Math object in JavaScript? Explain its four methods with example**

# **Explain with an example the purpose of following popup boxes: Alert, confirm, prompt**

# **Define DTD and its use. How to create Internal DID in XML.7 Give example.**

# **What is XML? Discuss its any four features**

# **Describe Document Object Model with diagram**

# **Write a JavaScript code to input a number from user into variable n, display table of factorials up to n**

# **Unit-3**

# **Write a program in Ajax to display the last access date of a text file.**

# **Explain any two methods of XMLHttpRequest object.**

# **How to create arrays in PHP? Explain with example**

# **Give if structures in PHP. Explain with an example.**

# **What is the use of session in PHP? Explain any three functions of it.**

# **Discuss Query methods for traversing**

# **Explain any two functions used for animation in Query**

# **Write a PHP code to check given number is positive, negative or Zero. The number is to be accepted from HTML form**

# **Explain the following jQuery traversing methods**

# **filter() end() contents() . find()**

# **What is the purpose of XMLHttpRequest object in AJAX? How to open it in AJAX? Explain.**

# **What is a PHP session? How to start, get, modify and destroy a PHP session? Give example.**

# **Explain the following functions w.rt database handling in PHP**

# **mysql\_connect()**

# **mysql tablename()**

# **mysql.ping0**

# **mysql\_drop\_db()**

# **mysql\_list\_fields()**

# **What is XML HupRequest object? Explain the following properties of**

# **XMLHttpRequest object**

# **onreadystatechange**

# **readyState**

# **responseText**

# **status**

# **What is a PHP session? How to start, get, modify and destroy a PHP session? Give example**

# **Define Query. Give its any three features and basic syntax with an example**

# **How does AJAX handle asynchronous data transfer?**

# **Which are the methods for creating animation effects in Query? Explain any two methods with example.**

# **Write a PHP code to check number entered through HTML page is positive, negative or equal to zero. Display appropriate message.**

# **Mix**

# **What is the purpose of audio and video tags in HTML5? Give example**

# **Write JavaScript function to calculate age from the date of birth.**

# **Give PHP code to create / retrieve a cookie, with following condition**

# **Cookie name should be 'demouser with values 'demo**

# **It should expire after 10 days.**

# **Explain the following elements in XSLT**

# **xsl:template**

# **xsl:import**

# **Discuss various features of AJAX**

# **Write a code using PHP and HTML to accept department id from user and display department name and address in a table.**

# **[Table: Department {Deptid number (3), Deptname varchar (20), Deptaddress varchar2 (10)}]**

# **Diagrammatically explain AJAX Web Application Model.**

# **How does AJAX handle asynchronous data transfer?**

# **What is the purpose of following CSS properties?**

# **text-align**

# **font-style**

# **border**

# **font-size**

# **display**

# **Explain any five PHP Comparison Operators.**

# **Explain the following table tags in HTML5 with example-**

# **<table>**

# **<tr>**

# **<th>**

# **<td>**

# **<table align =”right”>**

# **What is the use ofthe following CSS selectors? Explain with example:-**

# **.class**

# **#id**

# **What is the purpose of Window object in JavaScript? Explain its any two properties and methods with code snippet.**

# **Diagrammatically explain AJAX Web Application Model.**

# **Write a PHP code to create an array of five student names and display its values using foreach loop.**