COMPUTER MULTIMEDIA & ANIMATIONS

IV SEM -BCA

UNIT –I

2 mark questions:

1.Differentiate HTML and XHTML:

HTML (HyperText Markup Language) is a markup language used for creating the structure and presentation of web pages. It has more lenient syntax rules and allows for more flexibility in writing code. It supports optional closing tags and allows for the use of both uppercase and lowercase tags.

XHTML (eXtensibleHyperText Markup Language) is a stricter version of HTML that follows the rules of XML. It requires all tags to be properly nested and closed, and all attribute values to be enclosed in quotes. XHTML has a more standardized syntax and is well-formed XML, making it easier for web browsers and other XML tools to parse and process.

2.Explain basic HTML markup (tag) syntax with a diagram:

Basic HTML markup consists of opening and closing tags, with content placed between them. Here's an example diagram showing the syntax:

<tagname> Content </tagname>

<tagname> represents the HTML element, such as <p> for a paragraph or <h1> for a heading.

Content refers to the text or other HTML elements contained within the tags.

3.The use of the <audio> tag:

The <audio> tag is used to embed audio content, such as music or sound clips, into an HTML document. Important attributes and their purposes include:

src: Specifies the URL of the audio file.

controls: Displays audio controls (play, pause, volume) to the user.

autoplay: Starts playing the audio automatically when the page loads.

loop: Causes the audio to repeat playback continuously.

preload: Specifies whether the audio should be loaded when the page loads.

4.The use of the <video> tag:

The <video> tag is used to embed video content into an HTML document. Important attributes and their purposes include:

src: Specifies the URL of the video file.

controls: Displays video controls (play, pause, seek) to the user.

autoplay: Starts playing the video automatically when the page loads.

loop: Causes the video to repeat playback continuously.

poster: Defines an image to be displayed as the video thumbnail before playback.

5.Writing HTML5 comments:

HTML5 comments can be written using the following syntax:

<!-- This is an HTML5 comment -->

6**.Writing HTML5 conditional comments:**

HTML5 no longer supports conditional comments. Conditional comments were specific to earlier versions of Internet Explorer and allowed developers to target specific versions of the browser. However, since HTML5 is meant to be standardized across browsers, conditional comments are no longer necessary or supported.

7.The use of the target attribute of the <a> element:

The target attribute specifies where to open the linked document when the user clicks on the link. Possible values include:

_blank: Opens the linked document in a new tab or window.

_self: Opens the linked document in the same tab or window (default).

_parent: Opens the linked document in the parent frame or window.

_top: Opens the linked document in the full body of the window.

8.HTML5 audio tag with its attributes:

The HTML5 <audio> tag is used to embed audio content. Some important attributes include:

src: Specifies the URL of the audio file.

controls: Displays audio controls to the user.

autoplay: Starts playing the audio automatically when the page loads.

loop: Causes the audio to repeat playback continuously.

preload: Specifies whether the audio should be loaded when the page loads.

**9.The
 tag and its clear attribute:**

The
 tag is a line break tag that forces a line break within a paragraph or block of text. It does not have a closing tag. The clear attribute is not valid for the
 tag. It is used with the
 tag in older versions of HTML to control the alignment of text around line breaks.

10.The purpose of the tag and its nowrap attribute:

The tag is a generic container used to group and style block-level elements. It is commonly used for layout and styling purposes. The nowrap attribute is not a valid attribute for the tag. It is used with the <td> tag in older versions of HTML to prevent line breaks within table cells.

11.The purpose of the <dl> and <dt> elements:

<dl> (Definition List) is used to create a list of term/definition pairs.

<dt> (Definition Term) is used to define the term in a definition list.

12.The use of the <fieldset> element:

The <fieldset> element is used to group related form elements together. It is commonly used to create a visual grouping and provide a semantic structure to form controls.

13.The use of the heading (<h1> to <h6>) tag:

The heading tags (<h1> to <h6>) are used to define different levels of headings in an HTML document. They represent the hierarchical structure of the content, with <h1> being the highest (most important) and <h6> being the lowest (least important) heading level.

14.The use of the <head> tag:

The <head> tag is used to define the head section of an HTML document. It contains meta-information about the document, such as the title, character encoding, linked stylesheets, JavaScript files, and more.

15.The use of the <header> tag:

The <header> tag is used to define the header section of a document or a section within a document. It typically contains introductory content, headings, logos, and navigation menus.

16.The use of the <label> tag and its for attribute:

The <label> tag is used to create a label for form controls, such as input fields or checkboxes. The for attribute is used to associate the <label> with a specific form control using its corresponding id attribute. This association improves accessibility and usability, as clicking the label will focus or activate the associated control.

17.The purpose of the element:

The element is an inline container used to group and apply styles to a specific section of text or inline elements. It does not add any semantic meaning to the content.

18.The use of cellpadding and cellspacing attributes in the <table> element:

cellpadding defines the space between the content of a cell and its border.

cellspacing defines the space between cells in a table.

19.The purpose of colspan and rowspan attributes of the <td> tag:

colspan defines the number of columns a table cell should span.

rowspan defines the number of rows a table cell should span.

20.The purpose of the <th> tag and four attributes:

The <th> tag represents a header cell in a table.

Four attributes: colspan (spanning columns), rowspan (spanning rows), scope (defines the scope of the header cell), and abbr (defines an abbreviated version of the header cell content).

21.Four features of the <title> tag:

It defines the title of an HTML document that appears in the browser's title bar or tab.

It is essential for search engine optimization (SEO).

It helps users identify the page when bookmarking or saving it.

It is displayed in search engine results.

22.The use of the <ul> tag and the purpose of its type attribute:

The <ul> tag is used to create an unordered (bulleted) list.

The type attribute is not a valid attribute for the <ul> tag. It is used with the <ol> (ordered list) tag to specify the type of numbering or bullet style.

23.Ways to include JavaScript in a webpage:

Inline: JavaScript code can be written directly within <script> tags in the HTML document.

Example: <script>alert("Hello, World!"); </script>

External file: JavaScript code can be stored in a separate .js file and included in the HTML document using the <script> tag with the src attribute.

Example: <script src="script.js"></script>

24.The use of the <noscript></noscript> tag:

The <noscript></noscript> tag is used to provide alternative content that is displayed when JavaScript is disabled or not supported by the user's browser. The content within the <noscript> tags is rendered when JavaScript is unavailable.

25.Variable definition in JavaScript:

In JavaScript, variables can be declared using the var, let, or const keyword. Here's an example:

var age = 25;

26.Anonymous function in JavaScript:

An anonymous function is a function without a specified name. It can be assigned to a variable or used as a callback function. Example:

var greet = function() {

console.log("Hello!");

};

greet();

27.Uses of === and !== operators in JavaScript:

=== (strict equality operator) compares the equality of both value and type.

!== (strict inequality operator) compares inequality of both value and type. These operators are used to ensure a strict comparison without any type coercion.

28.Four special operators of JavaScript:

typeof: Returns the data type of a variable or expression.

instanceof: Checks if an object is an instance of a particular class.

delete: Deletes a property from an object or removes an element from an array.

in: Checks if a property exists in an object.

29.**Purpose of for...in and for...each loops in JavaScript:**

for...in loop iterates over the properties of an object.

for...each loop (also known as for...of loop) iterates over the iterable elements of an array, string, or other iterable objects.

30.Event and event handler:

An event is an action or occurrence, such as a button click, mouse movement, or keypress, that happens within the browser window.

An event handler is a piece of code (function) that is executed in response to a specific event. It defines the behavior or actions to be taken when the event occurs.

31.The use of the attachEvent() method:

The attachEvent() method is a legacy method used in older versions of Internet Explorer to attach an event handler to an element. Example:

```
element.attachEvent('onclick', function() {
  // Event handler code
});
```

32.Object initializers in JavaScript:

Object initializers provide a way to create and initialize objects with properties and values in a single statement. Example:

```
var person = {
name: "John",
age: 30,
city: "New York"
};
```

4/6 Mark Questions:

1.List and explain some of the most commonly used (X)HTML elements:

<p>: Represents a paragraph of text.

<a>: Creates a hyperlink to another webpage or resource.

<img>: Inserts an image into the document.

 and <ol>: Used to create unordered and ordered lists, respectively.

<table>: Defines a table structure to display tabular data.

: A generic container used for grouping and styling content.

2.HTML and XHTML DTD specifications:

DTD (Document Type Definition) specifies the rules and structure of an HTML or XHTML document. It defines the elements, attributes, and their relationship within a document. Example DTD declarations for HTML and XHTML:

HTML 5:

<!DOCTYPE html>

XHTML 1.0 Strict:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

3.HTML document structure:

<!DOCTYPE html>

<html>

<head>

<!-- Metadata and stylesheets -->

</head>

<body>

<!-- Content of the webpage -->

</body>

</html>

4.Elements inside HTML document head:

Common elements found inside the <head> tag include:

<title>: Specifies the title of the document.

<meta>: Defines metadata about the document.

<link>: Specifies external CSS stylesheets or other linked resources.

<style>: Contains embedded CSS styles for the document.

<script>: Includes JavaScript code or links to external JavaScript files.

5.(X)HTML rules:

Elements must have opening and closing tags, except for self-closing elements.

Tags and attribute names must be written in lowercase.

Attribute values must be enclosed in quotation marks.

Proper nesting of elements should be followed.

Elements should be used semantically and appropriately.

6.HTML5 structural elements:

HTML5 introduces new structural elements that provide better semantics for different parts of a webpage. Some of these elements include:

<header>: Represents the introductory content or a group of navigational elements at the top of a document or section.

<nav>: Defines a section containing navigation links.

<section>: Represents a standalone section of content within a document.

<article>: Represents a self-contained composition, such as a blog post or news article.

<aside>: Represents content that is tangentially related to the main content, like sidebars or pull quotes.

<footer>: Represents the footer of a document or section, containing metadata or copyright information.

7.Explain new HTML5 form fields.

HTML5 introduced several new form fields that provide improved functionality and user experience. Some of the new form fields include:

<input type="email">: Validates user input to ensure it matches the format of an email address.

<input type="url">: Validates user input to ensure it matches the format of a URL.

<input type="tel">: Displays a numeric keypad on mobile devices for entering phone numbers.

<input type="date">: Provides a date picker for selecting dates.

<input type="range">: Renders a slider control for selecting a value within a specified range.

<input type="color">: Displays a color picker for selecting a color.

<input type="search">: Renders a search input field with a clear button and search icon.

8.Short note on <!DOCTYPE> element:

The <!DOCTYPE> declaration specifies the document type and version being used in an HTML document. It is the first line of an HTML document and helps browsers interpret the document correctly. It informs the browser about the version of HTML or XHTML used and the rules to follow. For example, the HTML5 doctype declaration is <!DOCTYPE html>, which indicates that the document is an HTML5 document.

9.Use of <a> tag and 5 element-specific attributes:

The <a> tag is used to create hyperlinks that link to other webpages or resources. Some of the element-specific attributes of the <a> tag are:

href: Specifies the URL or destination of the link.

target: Defines where the linked document will open (e.g., in a new window or tab).

title: Provides a tooltip or additional information about the link.

download: Specifies that the target will be downloaded when clicked, rather than navigating to it.

rel: Indicates the relationship between the current document and the linked document, such as "nofollow" for search engine optimization.

10.Use of <article> element of HTML5 and five unique attributes:

The <article> element in HTML5 represents a self-contained composition or content that can be independently distributed or syndicated. It typically represents blog posts, news articles, or forum posts. Some unique attributes of the <article> element are:

hidden: Indicates that the article should be hidden from view.

lang: Specifies the language of the content within the <article> element.

role: Defines the role or type of the article.

aria-labelledby: Associates the article with an element that serves as its label.

itemid and itemprop: Used for structured data markup with microdata.

11.Short note on <aside> element with five unique attributes:

The <aside> element in HTML5 represents content that is tangentially related to the main content. It is often used for sidebars, pull quotes, or related links. Some unique attributes of the <aside> element are:

hidden: Indicates that the content should be hidden from view.

role: Defines the role or type of the aside content.

aria-labelledby: Associates the aside content with an element that serves as its label.

itemid and itemprop: Used for structured data markup with microdata.

Unique element-specific attributes of <body> tag:

12.Some unique attributes of the <body> tag in HTML are:

onload: Specifies a script to be executed when the document is loaded.

onunload: Defines a script to be executed when the document is unloaded.

bgcolor: Sets the background color of the document's body.

text: Specifies the default text color within the body.

link: Sets the color of hyperlinks.

vlink: Defines the color of visited hyperlinks.

13.Unique element-specific attributes of <button> tag:

Some unique attributes of the <button> tag in HTML are:

disabled: Disables the button, preventing user interaction.

form: Specifies the form the button belongs to.

formaction: Overrides the form's action attribute for this specific button.

autofocus: Sets the button to receive focus automatically when the page loads.

type: Specifies the type of button (e.g., "submit", "reset", "button").

14.Unique element-specific attributes of <font> tag:

Please note that the <font> tag has been deprecated in HTML5, and it is recommended to use CSS for styling instead. However, some unique attributes of the <font> tag were:

color: Sets the color of the text.

size: Defines the size of the font.

face: Specifies the font family or typeface to be used.

15.<form> tag and its element-specific attributes:

The <form> tag in HTML is used to create a form that contains input elements for collecting user data. Some element-specific attributes of the <form> tag are:

action: Specifies the URL or file name to which the form data will be submitted.

method: Defines the HTTP method to be used when submitting the form (e.g., "GET" or "POST").

enctype: Specifies how form data should be encoded and submitted to the server.

target: Defines where the response from the form submission should be displayed (e.g., in a new window or frame).

autocomplete: Specifies whether the browser should autocomplete input fields within the form.

16.<hr> tag and its element-specific attributes:

The <hr> tag in HTML is used to create a horizontal rule, which represents a thematic break in the content. Some element-specific attributes of the <hr> tag are:

size: Specifies the height or thickness of the rule.

color: Sets the color of the rule.

width: Defines the width of the rule.

align: Aligns the rule horizontally within its containing element.

17.Element-specific attributes of <html> tag:

Some element-specific attributes of the <html> tag in HTML are:

lang: Specifies the primary language of the document.

xmlns: Defines the XML namespace for the document.

manifest: Specifies the URL of the cache manifest file for offline web applications.

prefix: Defines the prefix for custom data attributes.

18.<iframe> element and its element-specific attributes:

The <iframe> tag is used to embed another HTML document within the current document. Some element-specific attributes of the <iframe> tag are:

src: Specifies the URL of the embedded document.

width and height: Sets the width and height of the iframe.

sandbox: Restricts the capabilities of the embedded content for security purposes.

allowfullscreen: Specifies whether the iframe can be displayed in fullscreen mode.

seamless: Indicates that the iframe should have no visible borders or scrollbars.

19.<img> tag and its element-specific attributes:

The <img> tag is used to insert an image into an HTML document. Some element-specific attributes of the <img> tag are:

src: Specifies the URL or file path of the image.

alt: Provides alternative text that describes the image.

width and height: Sets the width and height of the image.

title: Specifies additional information or a tooltip for the image.

loading: Specifies the image loading strategy (e.g., "lazy" for lazy loading).

20.<input> tag and its element-specific attributes:

The <input> tag is used to create various types of input fields within a form. Some element-specific attributes of the <input> tag are:

type: Specifies the type of input field (e.g., "text", "checkbox", "radio").

name: Associates a name with the input field, which is used when submitting form data.

value: Sets the initial value of the input field.

placeholder: Provides a short hint or example text within the input field.

required: Specifies that the input field must be filled out before submitting the form.

21. tag and its attributes:

The <li> tag is used to define a list item within an ordered (<ol>) or unordered (<ul>) list. Some attributes of the <li> tag are:

value: Specifies the value of the list item. This attribute is commonly used with ordered lists (<ol>).

type: Defines the type of marker used for the list item. This attribute is commonly used with ordered lists (<ol>).

class: Assigns a class name to the list item for styling or JavaScript purposes.

id: Provides a unique identifier for the list item, which can be used for targeting with CSS or JavaScript.

22.<link> tag and its attributes:

The <link> tag is used to link external resources, such as stylesheets, to an HTML document. Some attributes of the <link> tag are:

href: Specifies the URL or file path of the linked resource.

rel: Defines the relationship between the current document and the linked resource (e.g., "stylesheet" for CSS files).

type: Specifies the MIME type of the linked resource.

media: Specifies the media type or media query for which the linked resource is intended.

sizes: Provides hints to the browser about the sizes of the linked resource for responsive design.

23.<marquee> tag and its attributes:

The <marquee> tag is used to create a scrolling or moving text or image within an HTML document. Some attributes of the <marquee> tag are:

behavior: Defines the behavior of the marquee (e.g., "scroll", "slide", "alternate").

direction: Specifies the direction of the marquee (e.g., "left", "right", "up", "down").

scrollamount: Sets the speed of the scrolling marquee.

scrolldelay: Defines the delay between each scroll movement.

loop: Specifies the number of times the marquee should repeat (-1 for infinite).

24. tag (ordered list) and its attributes:

The <ol> tag is used to create an ordered list, where list items are numbered. Some attributes of the <ol> tag are:

type: Specifies the type of numbering used in the list (e.g., "1" for decimal, "A" for uppercase letters, "a" for lowercase letters).

start: Defines the starting value of the list items.

reversed: Specifies that the numbering should be reversed.

compact: Reduces the spacing between list items.

25.<script> tag and its element-specific attributes:

The <script> tag is used to embed or reference JavaScript code within an HTML document. Some element-specific attributes of the <script> tag are:

src: Specifies the URL or file path of an external JavaScript file.

type: Defines the MIME type of the script content (e.g., "text/javascript", "module").

async: Specifies that the script should be executed asynchronously.

defer: Indicates that the script should be deferred until after the document has been parsed.

nonce: Provides a cryptographic nonce (number used once) for inline scripts as a security measure.

26.<select> tag and its attributes:

The <select> tag is used to create a dropdown list of options within an HTML form. Some attributes of the <select> tag are:

name: Associates a name with the select element, which is used when submitting form data.

multiple: Specifies that multiple options can be selected.

size: Sets the number of visible options in the dropdown list.

disabled: Disables the select element, preventing user interaction.

autofocus: Sets the select element to receive focus automatically when the page loads.

27.<style> tag and its element-specific attributes:

The <style> tag is used to define CSS styles within an HTML document. Some element-specific attributes of the <style> tag are:

type: Specifies the MIME type of the style content (e.g., "text/css").

media: Defines the media type or media query for which the styles are intended.

scoped: Limits the scope of the styles to the current element and its children.

title: Specifies a title or name for the style block, which can be used for targeting with JavaScript.

nonce: Provides a cryptographic nonce (number used once) for inline styles as a security measure.

28.<table> tag and its element-specific attributes:

The <table> tag is used to create a table structure within an HTML document. Some element-specific attributes of the <table> tag are:

border: Specifies the width of the table border.

width: Sets the width of the table.

cellpadding: Defines the padding between the cell content and cell borders.

cellspacing: Specifies the spacing between cells.

summary: Provides a summary or description of the table for accessibility purposes.

29.<time> element and its datetime attribute:

The <time> element in HTML5 is used to represent a specific time or date. It can be used to provide semantic meaning to date and time information. The datetime attribute of the <time> element is used to specify the date and/or time value. It follows the ISO 8601 format, such as "YYYY-MM-DD" for dates or "YYYY-MM-DDTHH:MM" for dates with time.

30.<video> element and its attributes:

The <video> element is used to embed a video or audio content within an HTML document. Some attributes of the <video> tag are:

src: Specifies the URL or file path of the video file.

controls: Displays native video controls (play, pause, volume, etc.).

autoplay: Specifies that the video should start playing automatically.

loop: Indicates that the video should replay after reaching the end.

poster: Sets an image to be displayed while the video is loading or before it starts.

31.JavaScript is a widely used programming language that is primarily used for adding interactivity and dynamic behavior to websites. It is a high-level, interpreted language that is executed on the client-side (in the user's web browser) and can interact with HTML and CSS to manipulate and modify web page content. Some key features of JavaScript include:

Dynamic and Interactive: JavaScript allows developers to create dynamic and interactive web pages by enabling changes to the content and behavior of web elements in response to user actions or events.

Client-Side Execution: JavaScript is executed on the client-side, meaning it runs within the user's web browser. This eliminates the need for server-side processing for certain functionalities and enhances the responsiveness of web applications.

Data Types: JavaScript supports various data types, including numbers, strings, booleans, objects, arrays, functions, and more. It also provides built-in methods and functions to manipulate and work with these data types.

Event-Driven Programming: JavaScript utilizes event-driven programming, where actions or events (such as a button click or page load) trigger specific JavaScript code to execute. This allows developers to create interactive and responsive web applications.

DOM Manipulation: JavaScript provides powerful APIs for manipulating the Document Object Model (DOM) of an HTML document. This allows developers to dynamically modify the structure, content, and styles of web elements in real-time.

Support for Libraries and Frameworks: JavaScript has a vast ecosystem of libraries and frameworks, such as React, Angular, and Vue.js, which provide additional tools and abstractions to simplify web development and enhance productivity.

32.JavaScript supports several data types, including:

Number: Represents numeric values, both integers and floating-point numbers. Example: let age = 25;

String: Represents a sequence of characters enclosed in single or double quotes. Example: let name = "John";

Boolean: Represents either true or false. Example: let isActive = true;

Object: Represents a collection of key-value pairs or properties. Example: let person = { name: "John", age: 25 };

Array: Represents an ordered list of values. Example: let numbers = [1, 2, 3, 4, 5];

Null: Represents the absence of any value. Example: let value = null;

Undefined: Represents an uninitialized variable or a variable without a value assigned. Example: let variable;

unction: Represents a reusable block of code that can be invoked or called. Example: function sayHello() { console.log("Hello!"); }

33.

In JavaScript, a function is defined using the function keyword followed by the function name, parentheses containing optional parameters, and curly braces {} enclosing the function body. Here's an example:

```
function greet(name) {

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

}


// Calling the function

greet("John"); // Output: Hello, John!
```

In this example, the function greet takes a parameter name and logs a greeting message to the console using the parameter value.

In JavaScript, there are different ways to define functions:

Function Declaration: It is defined using the function keyword followed by the function name, parentheses containing optional parameters, and the function body enclosed in curly braces {}. Example:

```
function add(a, b) {

return a + b;

}
```

Function Constructor: It involves creating a new function using the Function constructor. It takes a string of comma-separated parameter names and the function body as arguments. Example:

var multiply = new Function("a", "b", "return a * b;");

Function Expression: It assigns a function to a variable or constant. The function is defined using the function keyword, followed by optional parameters and the function body, which is assigned to the variable. Example:

var subtract = function(a, b) {

return a - b;

};

35.

Event handlers in JavaScript are used to associate JavaScript code with specific events that occur on HTML elements. Event handlers can be added to HTML elements using attributes or by registering them through JavaScript code. Here's an example using an attribute:

<button onclick="handleClick()">Click me</button>

<script>

functionhandleClick() {

alert("Button clicked!");

}

</script>

In this example, the onclick attribute is added to the <button> element, which specifies the JavaScript code to execute when the button is clicked. The handleClick() function is called, and an alert is displayed.

36. TheaddEventListener() method in JavaScript is used to attach an event listener to an HTML element. It allows for more flexible event handling and supports multiple event listeners for a single element. Here's an example:

var button = document.querySelector("button");

button.addEventListener("click", function() {

alert("Button clicked!");

});

In this example, the addEventListener() method is used to add a click event listener to the button element. When the button is clicked, the anonymous function is executed, and an alert is displayed.

37.A constructor function in JavaScript is used to create and initialize objects. It serves as a blueprint or template for creating multiple objects with similar properties and behaviors. Here's an example:

```
function Person(name, age) {
  this.name = name;
this.age = age;
}

// Creating objects using the constructor function
var person1 = new Person("John", 25);
var person2 = new Person("Jane", 30);
```

In this example, the Person constructor function is defined with name and age parameters. It sets the properties of the created object using the this keyword. Objects can be created using the new keyword followed by the constructor function.

38

The alert() and confirm() methods are built-in functions in JavaScript used to display dialog boxes to the user.

alert(message): Displays an alert dialog box with the specified message and an OK button. Example: alert("Hello!");

confirm(message): Displays a confirmation dialog box with the specified message, an OK button, and a Cancel button. It returns true if the user clicks OK and false if the user clicks Cancel. Example:

```
var result = confirm("Are you sure?");
if (result) {
console.log("User clicked OK");
```

```
} else {

console.log("User clicked Cancel");

}
```

39.The prompt() method is a built-in function in JavaScript that displays a dialog box to the user with a message, an input field, and OK and Cancel buttons. It is used to get input from the user. The prompt() method returns the value entered by the user as a string or null if the user clicks Cancel. Example:

```
var name = prompt("Please enter your name:");

if (name) {

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

} else {

console.log("No name entered.");

}
```

UNIT –II

2 Mark questions:

1.

CSS stands for Cascading Style Sheets. It is a styling language used to describe the presentation and appearance of a document written in HTML or XML. CSS is used to define the layout, colors, fonts, and other visual aspects of web pages. It provides a way to separate the content of a web page from its presentation, making it easier to maintain and update the design of a website.

2.

 and are both HTML tags that are commonly used in conjunction with CSS for styling purposes.

: The tag is a block-level element that is used to group and create divisions or sections in an HTML document. It does not have any semantic meaning on its own but provides a container that can be styled using CSS. It is often used to structure the layout of a web page and apply styles to a group of elements.

: The tag is an inline element that is used to apply styles to a specific section of text or a small part of an HTML document. It does not affect the document's structure and is commonly used to apply CSS styles or add hooks for JavaScript interactions to a specific portion of text.

3.Some tags to avoid from HTML are:

: The tag was used in older versions of HTML to horizontally center-align content. However, it is deprecated in HTML5, and its functionality can be achieved using CSS instead.

<font>: The <font> tag was used to define font-related attributes, such as size, color, and face, directly within the HTML document. It is also deprecated in HTML5, and its functionality should be handled using CSS styles.

<b>, <i>, <u>: These tags were used for bold, italic, and underline formatting, respectively. However, it is considered better practice to use CSS styles (font-weight, font-style, text-decoration) to achieve these effects.

4.The main parts of a CSS style include:

Selectors: Selectors target specific HTML elements or groups of elements to which the style rules will be applied.

Properties: Properties define the visual characteristics of the selected elements, such as color, font size, margin, padding, etc.

Values: Values are assigned to properties and determine the specific appearance or behavior of the selected elements. They can be predefined keywords or custom values.

Declaration Block: The declaration block is enclosed in curly braces {} and contains one or more property-value pairs. It represents a set of style rules that will be applied to the selected elements.

5.There are three ways to use style sheets in an HTML document:

Inline Styles: Inline styles are applied directly to individual HTML elements using the style attribute. The CSS rules are defined within the attribute's value. Example: This is a blue paragraph.

Internal Stylesheets: Internal stylesheets are defined within the <style> tags in the <head> section of an HTML document. CSS rules are written between the <style> tags and apply to the entire document. Example:

```
<head>

<style>

p {

color: blue;

   }

</style>

</head>

<body>

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

</body>
```

External Stylesheets: External stylesheets are separate CSS files that are linked to the HTML document using the <link> tag. The CSS rules are defined within the external stylesheet file and can be applied to multiple HTML pages. Example:

```
<head>

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

</head>

<body>

This is a paragraph with styles applied from the external stylesheet.

</body>

6.

The <link> tag in HTML is used to link an external resource, such as a CSS file, to the HTML document. It is primarily used to connect external style sheets to HTML. The <link> tag has the following attributes:

rel: Specifies the relationship between the HTML document and the linked resource. For CSS stylesheets, the rel attribute should be set to "stylesheet".

href: Specifies the path or URL to the external resource, such as the CSS file.

type: Specifies the MIME type of the linked resource. For CSS stylesheets, the type attribute should be set to "text/css".

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

7.Inline styles are CSS styles that are directly applied to individual HTML elements using the style attribute. The style attribute is used to define one or more CSS property-value pairsinline with the HTML element. Here's an example:

This is a paragraph with inline styles.

In this example, the style attribute is added to the element, and the CSS styles are defined within the attribute's value. The color property is set to "blue" and the font-size property is set to "16px".

8.Different types of selectors in CSS include:

Element Selector: Selects elements based on their HTML tag name. Example: p { color: blue; } targets all elements.

Class Selector: Selects elements based on the value of their class attribute. Example: .highlight { background-color: yellow; } targets elements with class="highlight".

ID Selector: Selects a single element based on its id attribute. Example: #header { font-size: 24px; } targets the element with id="header".

Attribute Selector: Selects elements based on their attribute values. Example: input[type="text"] { border: 1px solid black; } targets <input> elements with type="text".

Pseudo-class Selector: Selects elements based on a specific state or condition. Example: a:hover { color: red; } targets <a> elements when hovered over.

9.The universal selector (*) in CSS selects all elements in an HTML document. It can be used to apply styles globally or as a wildcard when combined with other selectors. Its use includes:

Applying a common style to all elements: * { margin: 0; padding: 0; } sets the margin and padding of all elements to zero.

Resetting default styles: * { font-family: Arial, sans-serif; } applies the Arial font to all elements.

10. The :not selector in CSS is used to select elements that do not match a specific selector. It allows excluding elements from a selection. Example: p:not(.highlight) { color: red; } selects all <p> elements except those with the class "highlight" and applies a red color to them.

11.To use web fonts in CSS, two CSS commands are required:

@font-face: This CSS rule is used to specify a custom font and its source. It allows the browser to download and use the font for rendering text. Example:

@font-face {

font-family: 'CustomFont';

src: url('font.woff2') format('woff2'),

url('font.woff') format('woff');

}

font-family: This CSS property is used to specify the font-family for an element. Example: body { font-family: 'CustomFont', Arial, sans-serif; } sets the font-family of the <body> element to 'CustomFont', and if it's not available, it falls back to Arial and sans-serif.

12.To make a font bold, the font-weight property can be used and set to bold. Example: font-weight: bold;

To make a font italic, the font-style property can be used and set to italic. Example: font-style: italic;

13.There are different ways to specify colors in CSS:

Keywords: CSS provides predefined color keywords such as red, blue, green, etc. Example: color: red;

Hexadecimal: Colors can be represented using a six-digit hexadecimal value. Example: color: #FF0000;

RGB: Colors can be defined using the RGB color model. Example: color: rgb(255, 0, 0);

RGBA: RGBA allows specifying an additional alpha value for transparency. Example: color: rgba(255, 0, 0, 0.5);

HSL: Colors can be defined using the HSL color model, specifying the hue, saturation, and lightness. Example: color: hsl(0, 100%, 50%);

HSLA: HSLA allows specifying an additional alpha value for transparency. Example: color: hsla(0, 100%, 50%, 0.5);

14. To capitalize text in CSS, the text-transform property can be used with the value capitalize. Example: text-transform: capitalize; This transforms the first character of each word to uppercase and the rest to lowercase.

16. The text-decoration property in CSS is used to add visual effects to text. It can be used to underline, overline, strike through, or style text with a line. Example: text-decoration: underline; This underlines the text.

17.The text-shadow property in CSS is used to add shadow effects to text. The values of the text-shadow property specify the color, horizontal offset, vertical offset, and optional blur radius of the shadow. Example: text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5); This adds a black shadow with a 2-pixel horizontal offset, 2-pixel vertical offset, and 4-pixel blur radius.

18.CSS provides several properties for aligning text:

text-align: Specifies the horizontal alignment of text within its container. Example: text-align: center; centers the text.

vertical-align: Specifies the vertical alignment of inline elements within a line. Example: vertical-align: middle; vertically aligns the inline elements to the middle.

line-height: Specifies the height of a line of text. Example: line-height: 1.5; sets the line height to 1.5 times the font size.

19.Margin and padding can be specified using shorthand notation:

Margin shorthand: margin property allows specifying margin values for all four sides in one declaration. Example: margin: 10px 20px 10px 20px; sets the top margin to 10px, right margin to 20px, bottom margin to 10px, and left margin to 20px.

Padding shorthand: padding property allows specifying padding values for all four sides in one declaration. Example: padding: 10px 20px 10px 20px; sets the top padding to 10px, right padding to 20px, bottom padding to 10px, and left padding to 20px.

20.Inline boxes and block boxes are two types of boxes in CSS:

Inline Boxes: Inline boxes are used to contain inline-level elements, such as text or inline elements like . They do not create line breaks and are stacked horizontally. Their dimensions are determined by the content inside them.

Block Boxes: Block boxes are used to contain block-level elements, such as or <p>. They create line breaks and occupy the entire width available by default. Their dimensions can be controlled using CSS properties like width and height.

21.The border property in CSS can be specified using shorthand notation:

border: Sets the width, style, and color of all four borders in one declaration. Example: border: 1px solid black; sets a 1px solid black border on all sides.

Background color can be provided in CSS using the background-color property. Example: background-color: red; sets the background color to red.

22.The box-shadow property in CSS is used to add a shadow effect to an element's box. It specifies the color, horizontal offset, vertical offset, blur radius, and spread radius of the shadow. Example: box-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5); This adds a black shadow with a 2-pixel horizontal offset, 2-pixel vertical offset, 4-pixel blur radius, and default spread radius.

23.The float property in CSS is used to specify whether an element should float to the left or right of its container. It is often used for creating layouts with multiple columns or to wrap text around images. The options for the float property are:

left: The element floats to the left side.

right: The element floats to the right side.

none: The element does not float and appears in the normal flow of the document.

inherit: The element inherits the float behavior from its parent.

Example: float: left; makes the element float to the left side.

24.Animation in CSS allows elements to gradually change their style over time. The start and end states define the initial and final appearance of the animated element.

Start State: The start state defines the initial appearance of the element before the animation starts. It includes the styles that will be animated.

End State: The end state defines the final appearance of the element after the animation completes. It includes the styles that the element will have at the end of the animation.

25.

Keyframes in CSS are used to define the intermediate states of an animation. They specify the style properties and their values at specific points in time during the animation. Keyframes are defined using the @keyframes rule and can be assigned a name. The animation properties refer to these keyframes to create a smooth transition between the specified states.

26.The animation-direction property in CSS specifies whether an animation should play in reverse or alternate directions. It has the following values:

normal: The animation plays in the default forward direction.

reverse: The animation plays in the reverse direction.

alternate: The animation plays forward, then reverses, creating an alternating effect.

alternate-reverse: The animation plays in reverse, then forwards, creating an alternating effect.

27.The shorthand syntax of the animation property in CSS is as follows:

animation: name duration timing-function delay iteration-count direction;

name: Specifies the name of the keyframes rule to be applied.

duration: Specifies the time it takes to complete one cycle of the animation.

timing-function: Specifies the acceleration curve for the animation.

delay: Specifies the delay before the animation starts.

iteration-count: Specifies the number of times the animation should repeat.

direction: Specifies the direction of the animation.

Example: animation: myAnimation 2s ease-in-out 1s infinite alternate;

28.

CSS transitions define the properties that will change smoothly over a specified duration when a state change occurs. The properties defined by CSS transitions include:

transition-property: Specifies the CSS properties to which the transition effect should be applied.

transition-duration: Specifies the duration of the transition effect.

transition-timing-function: Specifies the acceleration curve for the transition.

transition-delay: Specifies the delay before the transition effect starts.

28.The values for the transition-timing-function property in CSS define the acceleration curve for the transition. Some common values include:

ease: The transition starts slowly, accelerates in the middle, and slows down at the end (default).

linear: The transition progresses linearly at a constant speed.

ease-in: The transition starts slowly and accelerates.

ease-out: The transition slows down and ends gradually.

ease-in-out: The transition starts slowly, accelerates, and then slows down.

4/6 Mark Questions:

1. Advantages of CSS:

Separation of Presentation and Content: CSS allows separating the presentation of a web page from its content. This separation makes it easier to maintain and update the styling of a website without modifying the HTML structure.

Consistency: CSS enables consistent styling across multiple web pages by applying the same stylesheets to all pages. Changes made in one place can be automatically reflected on all pages.

Improved Page Loading and Performance: CSS files can be cached by the browser, reducing the amount of data that needs to be downloaded for subsequent page visits. This improves page loading speed and overall performance.

Flexibility and Control: CSS provides precise control over the layout, positioning, and styling of HTML elements. It offers a wide range of selectors and properties to target specific elements and apply custom styles.

Responsive Design: CSS facilitates creating responsive web designs that adapt to different screen sizes and devices. Media queries can be used to apply different styles based on the viewport size, enabling a consistent user experience across devices.

Accessibility: CSS allows developers to improve the accessibility of web content by separating the structure from the visual presentation. This makes it easier to apply appropriate accessibility features and ensure compliance with accessibility standards.

2.Style elements of CSS refer to the properties and values that define the visual appearance and layout of HTML elements. Some common style elements in CSS include:

color: Sets the color of the text.

background-color: Sets the background color of an element.

font-family: Specifies the font family or typeface to be used for text.

font-size: Sets the size of the font.

font-weight: Specifies the thickness or boldness of the font.

text-align: Aligns the text horizontally within its container.

text-decoration: Adds decorations to the text, such as underline or strikethrough.

padding: Sets the space between the content and the border of an element.

margin: Sets the space outside the border of an element.

border: Defines the style, width, and color of the border around an element.

display: Specifies how an element is displayed (e.g., block, inline, or none).

position: Controls the positioning of an element on the page (e.g., relative, absolute, or fixed).

These are just a few examples of the style elements in CSS. CSS offers a wide range of properties and values to control various aspects of the visual presentation and layout of web pages.

3.Different ways to link style sheets in HTML5:

External Style Sheet: Linking an external CSS file using the <link> element in the HTML <head> section. Example: <link rel="stylesheet" href="styles.css">

Internal Style Sheet: Including the CSS code directly within the HTML file using the <style> element in the HTML <head> section. Example: <style> /* CSS code here */ </style>

Inline Styles: Applying styles directly to individual HTML elements using the style attribute. Example: This is a blue paragraph.

4. Steps for creating an internal style sheet:

Open an HTML file in a text editor or an HTML editor.

Locate the <head> section of the HTML file.

Within the <head> section, add the <style> element to define the internal style sheet.

Inside the <style> element, write the CSS code to specify the desired styles.

Save the HTML file.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Internal Style Sheet</title>
<style>
body {
background-color: lightblue;
font-family: Arial, sans-serif;
    }
    h1 {
color: red;
text-align: center;
    }
</style>
</head>
<body>
<h1>Welcome to My Website</h1>
<p>This is some sample text.</p>
</body>
```

</html>

In the above example, the <style> element is used to define an internal style sheet within the <head> section of the HTML document. The CSS code inside the <style> element applies specific styles to the <body> and <h1> elements.

5.Steps for creating an external style sheet:

Create a new CSS file with a .css extension. For example, styles.css.

Open the CSS file in a text editor or a CSS editor.

Write the desired CSS rules and styles in the CSS file.

Save the CSS file.

In the HTML file, link the external style sheet using the <link> element in the <head> section. Use the rel attribute with the value "stylesheet" and the href attribute with the value pointing to the CSS file.

Example:

styles.css:

```
body {

background-color: lightblue;

font-family: Arial, sans-serif;

}


h1 {

color: red;

text-align: center;

}
```

index.html:

```
<!DOCTYPE html>

<html>
```

<head>

<title>External Style Sheet</title>

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

</head>

<body>

<h1>Welcome to My Website</h1>

This is some sample text.

</body>

</html>

In the above example, the CSS styles are written in an external CSS file named styles.css. The HTML file (index.html) includes the external style sheet by linking it using the <link> element in the <head> section. The href attribute specifies the path to the CSS file.

This approach allows for the separation of the CSS code from the HTML file, making it easier to maintain and update the styles independently.

The <link> tag in HTML is used to link an external resource, such as a style sheet, to an HTML document. It is commonly used to link CSS files. The <link> tag has the following attributes:

rel: Specifies the relationship between the current document and the linked resource. In the case of a style sheet, the value should be set to "stylesheet".

href: Specifies the URL or path to the external resource (e.g., the CSS file).

type: Specifies the MIME type of the linked resource. For CSS, the value should be set to "text/css".

media: Specifies the media type or media query for which the linked resource should be applied. It allows for conditional styling based on the device or screen size. Common values include "all" (default), "screen", "print", "handheld", etc.

Example:

<!DOCTYPE html>

<html>

```
<head>

<title>Linking CSS File</title>

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

</head>

<body>

<h1>Welcome to My Website</h1>

<p>This is some sample text.</p>

</body>

</html>
```

In the above example, the <link> element is used to link an external CSS file named styles.css. The href attribute specifies the path to the CSS file, and the type attribute indicates that it is a CSS file. The media attribute is set to "screen", indicating that the styles should be applied when viewing the web page on a screen.

Inline style refers to applying CSS styles directly within an HTML element using the style attribute. It allows you to define specific styles for individual elements without the need for a separate CSS file. Here's an example:

html

Copy code

```
<!DOCTYPE html>

<html>

<head>

<title>Inline Style Example</title>

</head>

<body>

<h1 style="color: blue; font-family: Arial, sans-serif;">Welcome to My Website</h1>

<p style="color: red;">This is some sample text.</p>

</body>

</html>
```

In the above example, the style attribute is used within the <h1> and <p> elements to apply inline styles. The styles defined within the style attribute override any external or internal styles.

Different types of selectors in CSS:

Element Selector: Selects elements based on their tag name. For example, h1 selects all <h1> elements.

Class Selector: Selects elements based on their class attribute. It is denoted by a dot (.) followed by the class name. For example, .highlight selects all elements with class="highlight".

ID Selector: Selects elements based on their ID attribute. It is denoted by a hash (#) followed by the ID name. For example, #logo selects the element with id="logo".

Attribute Selector: Selects elements based on their attribute values. It can be used to select elements with specific attributes or attribute values. For example, [type="submit"] selects all elements with type="submit".

Pseudo-class Selector: Selects elements based on their state or position in the document. Pseudo-classes are denoted by a colon (:) followed by the pseudo-class name. For example, :hover selects an element when it is being hovered over.

The universal selector (*) in CSS selects all elements in the document. It is useful when you want to apply a style to every element on the page. It is denoted by an asterisk (*). For example, * { margin: 0; padding: 0; } sets the margin and padding of all elements to zero.

The universal selector can be useful for resetting or normalizing styles across different browsers and elements.

The :not selector in CSS is used to select elements that do not match a specific selector. It is denoted by :not(selector). For example, p:not(.special) selects all <p> elements that do not have the class "special".

The :not selector allows you to apply styles to elements that meet certain criteria while excluding others. It can be useful for targeting specific elements and applying different styles to them.

12.Two CSS commands required to use web fonts are @font-face and font-family:

@font-face: This CSS at-rule is used to define custom fonts that can be used in a web page. It specifies the font file's location and assigns a name to the font face.

Example:

@font-face {

font-family: MyCustomFont;

src: url('path/to/font.woff2') format('woff2'),

url('path/to/font.woff') format('woff');

}

font-family: This CSS property is used to specify the font family or font stack to be used for an element. The value should match the name given in the @font-face rule.

Example:

h1 {

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

}

In the above example, @font-face defines a custom font named "MyCustomFont" and specifies the font file paths. The font-family property is then used to apply the custom font to the <h1> element, falling back to Arial and sans-serif if the custom font is not available.

To use the @font-face directive in CSS, follow these steps:

Define the @font-face rule with the font-family and src properties.

Specify the font file's location using the src property. Multiple formats can be provided for better browser compatibility.

Assign a name to the font face using the font-family property.

Use the defined font family name in the font-family property of any element to apply the custom font.

Example:

@font-face {

font-family: MyCustomFont;

```
src: url('path/to/font.woff2') format('woff2'),
url('path/to/font.woff') format('woff');
}

h1 {
font-family: MyCustomFont, Arial, sans-serif;
}
```

In the above example, the @font-face rule defines a custom font named "MyCustomFont" and specifies the font file paths. The h1 element then uses the font-family property to apply the custom font, falling back to Arial and sans-serif if the custom font is not available.

The @font-face directive allows you to use custom fonts in your web pages, expanding the range of available typography options.

13. Filters are used in searching Google Web Fonts to refine and narrow down the results based on specific criteria. Here are some commonly used filters in Google Web Fonts:

Category: Allows you to filter fonts based on their category, such as serif, sans-serif, display, handwriting, etc.

Language: Filters fonts based on the supported language(s). You can select one or multiple languages to find fonts that support specific language characters.

Thickness: Lets you filter fonts based on their thickness or stroke width, ranging from thin to ultra-bold.

Slant: Allows you to filter fonts based on their slant or italic style, ranging from normal to italic.

Width: Filters fonts based on their width, ranging from condensed to expanded.

Script: Lets you filter fonts based on their script type, such as Latin, Cyrillic, Greek, etc.

Sort: Allows you to sort the search results based on popularity, date added, trending, or alphabetical order.

By using these filters, you can narrow down your search and find the most suitable fonts for your design needs, based on specific categories, languages, styles, and more.

14.There are different ways to set the color property value in CSS:

Color Name: You can use predefined color names to specify the color, such as "red", "blue", "green", etc. For example: color: red;

Hexadecimal Notation: Hexadecimal values represent colors using a combination of six characters, consisting of numbers (0-9) and letters (A-F). Each pair of characters represents the intensity of red, green, and blue (RGB) color channels. For example: color: #FF0000; (represents red color).

RGB Notation: RGB values allow you to define the intensity of red, green, and blue color channels using decimal numbers (0-255). For example: color: rgb(255, 0, 0); (represents red color).

RGBA Notation: RGBA values are similar to RGB values, but with an additional alpha channel that specifies the opacity (transparency) of the color. The alpha value ranges from 0 (fully transparent) to 1 (fully opaque). For example: color: rgba(255, 0, 0, 0.5); (represents semi-transparent red color).

HSL Notation: HSL (Hue, Saturation, Lightness) values allow you to define the color based on its hue, saturation, and lightness. The hue value ranges from 0 to 360, saturation and lightness values range from 0% to 100%. For example: color: hsl(0, 100%, 50%); (represents red color).

HSLA Notation: HSLA values are similar to HSL values, but with an additional alpha channel for transparency. For example: color: hsla(0, 100%, 50%, 0.5); (represents semi-transparent red color).

These different ways of setting the color property give you flexibility in defining colors for your elements in CSS.

15. The font-size property in CSS is used to specify the size of the font for an element. There are different ways to set the font-size property value:

Absolute Sizes: You can set the font size using absolute units like pixels (px), points (pt), inches (in), centimeters (cm), etc. For example: font-size: 16px;

Relative Sizes: Relative sizes are specified using relative units that are relative to the parent element or the default browser font size. Common relative units include percentages (%), em, and rem. For example: font-size: 120%;

Keyword Values: CSS also provides keyword values for font-size, such as small, medium, large, x-small, x-large, etc. These keyword values have predefined sizes associated with them. For example: font-size: large;

Viewport Units: Viewport units (vw, vh, vmin, vmax) allow you to set font sizes relative to the size of the viewport. For example: font-size: 5vw; (represents 5% of the viewport width).

16. The styling of lists using CSS allows you to customize the appearance of HTML lists, such as unordered lists (<ul>) and ordered lists (<ol>). Here are some common CSS properties used to style lists:

list-style-type: This property specifies the type of marker or numbering style for the list items. It can take values like disc, circle, square for unordered lists, and decimal, lower-roman, upper-alpha for ordered lists.

list-style-image: This property allows you to use a custom image as the marker for list items. You can specify the URL of the image to be used as the marker.

list-style-position: This property controls the position of the marker relative to the list item's content. It can be set to inside (marker inside the content area) or outside (marker outside the content area).

list-style: This is a shorthand property that combines the list-style-type, list-style-image, and list-style-position properties into a single declaration.

By using these properties, you can customize the marker style, position, and appearance of lists to match your design preferences.

17. The properties that make up the box model in CSS define the spacing and dimensions of an element. The main properties that make up the box model are:

Content: The actual content of the element, such as text, images, or other elements.

Padding: The space between the content and the element's border. It can be set using the padding property and can have different values for each side (top, right, bottom, left).

Border: The border around the element's content and padding. It can be set using the border property and can have properties like border-width, border-style, and border-color.

Margin: The space outside the element's border, creating the gap between neighboring elements. It can be set using the margin property and can have different values for each side (top, right, bottom, left).

These properties work together to define the size and spacing of an element within the layout of a web page.

18. When formatting individual borders in CSS, you can use the border property along with specific properties to control each side of the border. Here's an example:

/* Formatting the left border */

border-left-width: 2px;

border-left-style: dashed;

border-left-color: red;

In this example, we are formatting the left border of an element with a width of 2 pixels, a dashed style, and a red color. Similarly, you can use border-top, border-right, and border-bottom properties to format the other sides of the border.

19.There are multiple ways to create rounded corners in CSS. Here are two common methods:

Using the border-radius property: This property allows you to round the corners of an element by specifying the radius of the curve. For example:

/* Rounded corners with equal radius */

border-radius: 10px;

/* Rounded corners with different horizontal and vertical radii */

border-radius: 10px 20px;

In the first example, all corners of the element will have a rounded radius of 10 pixels. In the second example, the top-left and bottom-right corners will have a radius of 10 pixels, while the top-right and bottom-left corners will have a radius of 20 pixels.

Using individual border-*-radius properties: This method allows you to set different border radii for each corner of the element. For example:

/* Rounded corners with individual radii */

border-top-left-radius: 10px;

border-top-right-radius: 20px;

border-bottom-right-radius: 30px;

border-bottom-left-radius: 40px;

In this example, each corner of the element will have a different radius, allowing you to create custom rounded corner shapes.

20. The box-sizing property in CSS3 controls how the total width and height of an element is calculated, including the content, padding, and border. It has the following options:

content-box: This is the default value and calculates the width and height of an element excluding padding and border. The total width of the element is content width + padding + border.

border-box: With this value, the width and height of an element includes padding and border. The total width of the element is content width, and the padding and border are included within it.

By using the box-sizing property, you can control the sizing behavior of elements and easily manage the space taken by content, padding, and border.

21.The overflow property in CSS3 controls how content that overflows the boundaries of an element is displayed. It has the following options:

visible: This is the default value, where the content is not clipped and may extend beyond the boundaries of the element.

hidden: With this value, the overflowing content is hidden and not visible outside the element's boundaries.

scroll: This value adds a scrollbar to the element, allowing the user to scroll and view the overflowing content.

auto: This value automatically adds a scrollbar to the element if necessary, based on the content overflow.

By using the overflow property, you can control how overflowing content is handled within an element.

22. The clear property in CSS3 is used to control the behavior of floated elements. It has the following options:

none: This is the default value, where elements are allowed to float on both the left and right sides.

left: With this value, the element will not allow floating elements on the left side.

right: With this value, the element will not allow floating elements on the right side.

both: This value clears floats on both the left and right sides.

By using the clear property, you can control the positioning and clearing of floated elements within the layout.

23.CSS animation allows you to animate elements and create interactive and dynamic effects. Here's a brief explanation of CSS animation with an example:

CSS animation involves defining keyframes that specify the intermediate steps of an animation and then applying those keyframes to an element. For example, let's create a simple animation that moves a element from left to right:

```
@keyframes slide-in {

from {

transform: translateX(-100%);

  }

to {

transform: translateX(0);

  }

}
```

```
/* Applying the animation to the element */

.my-element {

animation: slide-in 2s ease-in-out;

}
```

In this example, we define a keyframe animation called slide-in that specifies the initial position of the element (from) and the final position (to). We then apply the animation to the .my-element class, specifying a duration of 2 seconds and an ease-in-out timing function.

When the animation is triggered, the element will smoothly move from left to right over the specified duration.

25.CSS provides several properties related to animation. Here are some of the important animation properties:

animation-name: Specifies the name of the keyframe animation to be applied to the element.

animation-duration: Sets the duration of the animation, specifying how long it takes to complete one cycle.

animation-timing-function: Defines the timing function that determines the acceleration and deceleration of the animation.

animation-delay: Specifies a delay before the animation starts.

animation-iteration-count: Sets the number of times the animation should repeat.

animation-direction: Determines whether the animation plays in the normal direction, in reverse, or alternating between the two.

animation-fill-mode: Controls how the element's styles are applied before and after the animation.

26.The animation-fill-mode property in CSS specifies how the styles of an element are applied before and after the animation. It has the following possible values:

none: This is the default value, where the styles are not affected by the animation.

forwards: With this value, the element retains the computed values set by the last keyframe of the animation.

backwards: This value applies the styles defined in the first keyframe of the animation to the element before the animation starts.

both: This value combines the effects of both forwards and backwards, retaining the styles from the last keyframe before the animation and applying the styles from the first keyframe before the animation starts.

By using different values for animation-fill-mode, you can control how the element's styles are affected before and after the animation.

****

UNIT III

2 Mark questions:

1.SVG stands for Scalable Vector Graphics. It is a markup language used to describe two-dimensional vector graphics. SVG graphics are defined using XML-based syntax and can be scaled to any size without losing quality. SVG is widely supported by modern web browsers and is commonly used for creating interactive graphics, animations, and illustrations on the web.

2.The parameters of the SVG <rect> element and their purpose are as follows:

x: Specifies the x-coordinate of the top-left corner of the rectangle.

y: Specifies the y-coordinate of the top-left corner of the rectangle.

width: Specifies the width of the rectangle.

height: Specifies the height of the rectangle.

rx: Specifies the horizontal radius of the rounded corners (optional).

ry: Specifies the vertical radius of the rounded corners (optional).

Example: <rect x="50" y="50" width="200" height="100" rx="10" ry="10" />

3.The parameters of the SVG <circle> element and their purpose are as follows:

cx: Specifies the x-coordinate of the center of the circle.

cy: Specifies the y-coordinate of the center of the circle.

r: Specifies the radius of the circle.

Example: <circle cx="100" cy="100" r="50" />

4.The parameters of the SVG <ellipse> element and their purpose are as follows:

cx: Specifies the x-coordinate of the center of the ellipse.

cy: Specifies the y-coordinate of the center of the ellipse.

rx: Specifies the horizontal radius of the ellipse.

ry: Specifies the vertical radius of the ellipse.

Example: <ellipse cx="100" cy="100" rx="50" ry="30" />

5.The parameters of the SVG <line> element and their purpose are as follows:

x1: Specifies the x-coordinate of the start point of the line.

y1: Specifies the y-coordinate of the start point of the line.

x2: Specifies the x-coordinate of the end point of the line.

y2: Specifies the y-coordinate of the end point of the line.

Example: <line x1="50" y1="50" x2="200" y2="200" />

6.The advantages of SVG <tspan> element include:

Allows for precise control over the styling and positioning of individual text segments within an SVG text element.

Supports the use of different fonts, sizes, colors, and other text properties for each <tspan> element.

Enables the creation of multiline text by grouping multiple <tspan> elements together.

Provides the ability to animate and transform individual text segments separately.

7.The SVG stroke-linecap attribute is used to control the shape of the endpoints of a stroke in a path or shape element. The possible values for stroke-linecap are:

butt: The stroke ends abruptly at the endpoint, without any extension.

round: The stroke ends with a rounded endpoint, extending beyond the endpoint by half the stroke width.

square: The stroke ends with a squared-off endpoint, extending beyond the endpoint by half the stroke width.

8.The SVG stroke-dasharray attribute is used to create dashed or dotted lines. It specifies the pattern of dashes and gaps in the stroke. The attribute value is a list of comma-separated values, where each value represents the length of a dash or gap. For example, stroke-dasharray="5,3" creates a dashed line with a dash of length 5 and a gap of length 3.

9.SVG grouping allows multiple SVG elements to be grouped together as a single unit using the <g> element. It helps in organizing and manipulating related elements collectively. Grouping elements with <g> can simplify transformations, apply styles, and define attributes to multiple elements simultaneously.

10. The SVG viewBox attribute defines the position and dimensions of the SVG viewport. It specifies the coordinate system and aspect ratio of the SVG content. The viewBox attribute can be used with various SVG elements, including <svg>, <symbol>, <image>, and <use>, among others. It allows scaling and resizing of the SVG content to fit different viewport sizes while maintaining the aspect ratio.

4/6 Mark Questions:

1.Advantages of SVG (Scalable Vector Graphics):

Resolution independence: SVG graphics are resolution-independent, meaning they can be scaled to any size without losing quality. This makes them ideal for devices with varying screen sizes and resolutions.

Small file size: SVG files are typically smaller in size compared to raster image formats like JPEG or PNG, making them load faster and reducing bandwidth usage.

Scalability: SVG graphics can be scaled up or down without pixelation or distortion, ensuring crisp and clear visuals at any size.

Editability: SVG files are created using XML-based code, which makes them highly editable. The shapes, paths, and attributes can be modified using a text editor or SVG editing software.

Interactivity: SVG supports interactivity and animation, allowing you to create interactive elements and dynamic graphics.

Accessibility: SVG graphics can be easily styled and customized to meet accessibility standards, making them accessible to users with disabilities.

2.Different ways to view an SVG file:

Web browsers: Most modern web browsers have built-in support for displaying SVG files. You can simply open the SVG file in a browser, and it will render the graphics.

SVG viewers: There are dedicated SVG viewer applications available that allow you to view and interact with SVG files. These viewers often provide additional features and options for working with SVG graphics.

Integrated development environments (IDEs): Some IDEs, such as Adobe Dreamweaver, provide a live preview feature that allows you to view and edit SVG files within the development environment.

Image editing software: Many image editing tools, like Adobe Illustrator or Inkscape, support SVG file formats. You can open an SVG file in these software applications to view and edit the graphics.

3.Different ways of embedding SVG in HTML5:

Inline embedding: You can directly embed SVG code within an HTML file using the <svg> element. The SVG code is placed between the opening <svg> and closing </svg> tags.

<img> tag: You can use the <img> tag and set the source (src) attribute to the path of the SVG file. This method treats the SVG file as an image and displays it on the webpage.

<object> tag: The <object> tag allows you to embed an SVG file as an external resource. You specify the SVG file path using the data attribute within the <object> tag.

<embed> tag: Similar to the <object> tag, the <embed> tag is used to embed external resources, including SVG files. You specify the SVG file path using the src attribute within the <embed> tag.

CSS background: You can use CSS to set the background of an HTML element to an SVG image. This can be achieved using the background-image property with the url() function, providing the path to the SVG file.

(Note: The choice of embedding method depends on the specific requirements and flexibility needed for the SVG usage.)

4.SVG Path Commands:

SVG path commands are used to define the shape of a path in an SVG file. The commands consist of single-letter codes followed by optional parameters. The primary SVG path commands are as follows:

M/m: Move To command (absolute/relative)

L/l: Line To command (absolute/relative)

H/h: Horizontal Line To command (absolute/relative)

V/v: Vertical Line To command (absolute/relative)

C/c: Cubic Bezier Curve To command (absolute/relative)

S/s: Smooth Cubic Bezier Curve To command (absolute/relative)

Q/q: Quadratic Bezier Curve To command (absolute/relative)

T/t: Smooth Quadratic Bezier Curve To command (absolute/relative)

A/a: Elliptical Arc To command (absolute/relative)

Z/z: Close Path command

5.SVG Path Commands for Curves and Arcs:

Cubic Bezier Curve (C/c): This command is used to draw a cubic Bezier curve. It takes three sets of parameters: two control points and an endpoint.

Example: C x1 y1, x2 y2, x y or c dx1 dy1, dx2 dy2, dx dy

Quadratic Bezier Curve (Q/q): This command is used to draw a quadratic Bezier curve. It takes two sets of parameters: a control point and an endpoint.

Example: Q x1 y1, x y or q dx1 dy1, dx dy

Elliptical Arc (A/a): This command is used to draw an elliptical arc. It takes five parameters: two radii, an x-axis rotation, an arc flag, and a sweep flag.

Example: A rxry x-axis-rotation large-arc-flag sweep-flag x y or a rxry x-axis-rotation large-arc-flag sweep-flag dx dy

6.SVG Path Commands for Lines:

Line To (L/l): This command is used to draw a straight line from the current point to a specified point.

Example: L x y or l dx dy

Horizontal Line To (H/h): This command is used to draw a horizontal line from the current point to a specified x-coordinate.

Example: H x or h dx

Vertical Line To (V/v): This command is used to draw a vertical line from the current point to a specified y-coordinate.

Example: V y or v dy

7.The T command in SVG path drawing is used to draw a smooth quadratic Bezier curve. It's a shorthand version of the Q/q command. The T command automatically calculates the control point based on the reflection of the previous control point. Here's an example of using the T command:

Example: T x y or t dx dy

8.SVG <text> element is used to render text within an SVG graphic. It has various attributes to control the appearance and positioning of the text. Some of the important attributes of the <text> element are:

x: Specifies the x-coordinate of the starting position of the text.

y: Specifies the y-coordinate of the starting position of the text.

dx: Specifies the horizontal offset from the current text position.

dy: Specifies the vertical offset from the current text position.

text-anchor: Specifies the alignment of the text relative to the x-coordinate (start, middle, end).

font-family: Specifies the font family for the text.

font-size: Specifies the font size of the text.

font-weight: Specifies the font weight (normal, bold).

fill: Specifies the text color.

text-decoration: Specifies decorations such as underline, overline, or line-through.

9.SVG <textpath> element is used to align text along a path within an SVG graphic. It allows the text to flow along the specified path. Here's an example of using the <textpath> element:

<path id="path" d="M10 80 C 40 10, 65 10, 95 80 S 150 150, 180 80" />

<text>

<textpathxlink:href="#path">This is the text along a path</textpath>

</text>

10.Different transformations available in SVG are:

translate(): Translates the element by a specified amount in the x and y directions.

Example: transform="translate(tx, ty)"

rotate(): Rotates the element by a specified angle around a given point.

Example: transform="rotate(angle, cx, cy)"

scale(): Scales the element by a specified factor in the x and y directions.

Example: transform="scale(sx, sy)"

skewX(): Skews the element by a specified angle along the x-axis.

Example: transform="skewX(angle)"

skewY(): Skews the element by a specified angle along the y-axis.

Example: transform="skewY(angle)"

matrix(): Applies a 2D transformation matrix to the element.

Example: transform="matrix(a, b, c, d, e, f)"

11.Variations of SVG translate() function with parameters:

translate(tx): Translates the element horizontally by tx units.

translate(tx, ty): Translates the element by tx units horizontally and ty units vertically.

12.Variations of SVG rotate() function with parameters:

rotate(angle): Rotates the element by the specified angle around its center point.

rotate(angle, cx, cy): Rotates the element by the specified angle around the specified center point (cx, cy).

13.SVG scale() function with parameters:

scale(sx): Scales the element uniformly by the specified factor sx.

scale(sx, sy): Scales the element independently in the x and y directions by the factors sx and sy, respectively.

14.SVG skewing is performed using skewX() and skewY() functions:

skewX(angle): Skews the element along the x-axis by the specified angle.

skewY(angle): Skews the element along the y-axis by the specified angle.

15.SVG presentation attributes are used to style and control the visual presentation of SVG elements. Some of the common SVG presentation attributes include:

fill: Specifies the color to fill the shape or text.

stroke: Specifies the color of the stroke (outline) of the shape or text.

stroke-width: Specifies the width of the stroke.

opacity: Specifies the transparency level of the element.

stroke-dasharray: Specifies the pattern of dashes and gaps in a stroke.

stroke-linecap: Specifies the shape used to end the stroke line.

16.SVG color codes include several formats:

Hexadecimal: #RRGGBB (e.g., #FF0000 for red)

RGB: rgb(R, G, B) (e.g., rgb(255, 0, 0) for red)

RGB with alpha: rgba(R, G, B, A) (e.g., rgba(255, 0, 0, 0.5) for semi-transparent red)

HSL: hsl(H, S, L) (e.g., hsl(0, 100%, 50%) for red)

HSL with alpha: hsla(H, S, L, A) (e.g., hsla(0, 100%, 50%, 0.5) for semi-transparent red)

17.Rules of SVG color specification:

Hexadecimal and RGB colors should be specified in uppercase.

Hexadecimal colors must start with a pound (#) symbol.

RGB colors should have values ranging from 0 to 255.

HSL colors have specific ranges: H (0-360), S (0-100%), L (0-100%).

Alpha values range from 0 (transparent) to 1 (fully opaque).

18.Opacity in SVG controls the transparency level of an element or its parts. It accepts values from 0 to 1, where 0 is fully transparent and 1 is fully opaque. The opacity can be set using the opacity attribute or through CSS using the opacity property. Example: opacity="0.5" or style="opacity: 0.5;"

19.SVG Gaussian blur effect is used to create a blur or softening effect on an element. It simulates a blurring effect similar to that of an out-of-focus photograph. The filter element with the <feGaussianBlur> child element is used to apply the blur effect. Example:

<filter id="blur-effect">

<feGaussianBlurstdDeviation="3" />

</filter>

<circle cx="50" cy="50" r="30" fill="blue" filter="url(#blur-effect)" />

20.SVG drop shadow effect adds a shadow behind an element, giving it a three-dimensional appearance. The filter element with the <feDropShadow> child element is used to apply the drop shadow effect. Example:

<filter id="drop-shadow">

<feDropShadow dx="2" dy="2" stdDeviation="3" />

</filter>

<rect x="10" y="10" width="100" height="50" fill="red" filter="url(#drop-shadow)" />

21.Lighting effects in SVG allow you to simulate different lighting conditions on an element, creating highlights, shadows, and reflections. SVG provides various lighting effects, such as <feDiffuseLighting>, <feSpecularLighting>, and <fePointLight>, to control the lighting properties and create desired effects.

22.Linear Gradient in SVG is used to create a smooth transition of colors along a straight line. It is defined using the <linearGradient> element. The x1, y1, x2, and y2 attributes are used to specify the start and end points of the gradient. Example:

<linearGradient id="gradient" x1="0%" y1="0%" x2="100%" y2="0%">

<stop offset="0%" stop-color="red" />

<stop offset="100%" stop-color="blue" />

</linearGradient>

<rect x="10" y="10" width="200" height="100" fill="url(#gradient)" />

Radial Gradient in SVG is used to create a circular or elliptical gradient. It is defined using the <radialGradient> element. The cx, cy, and r attributes are used to define the center and radius of the gradient. Example:

<radialGradient id="gradient" cx="50%" cy="50%" r="50%">

<stop offset="0%" stop-color="red" />

<stop offset="100%" stop-color="blue" />

</radialGradient>

<circle cx="100" cy="100" r="50" fill="url(#gradient)" />

****

UNIT- IV

2 Mark Questions:

1.The <canvas> element in HTML is used to draw graphics dynamically using JavaScript. It provides a drawing surface on which you can create and manipulate graphics, animations, and images. The <canvas> element has the following attributes:

width: Specifies the width of the canvas drawing surface.

height: Specifies the height of the canvas drawing surface.

2.JavaScript code to create a canvas drawing context:

```
const canvas = document.createElement('canvas');

const context = canvas.getContext('2d');
```

3.Parameters of the canvas strokeRect() method:

x: The x-coordinate of the top-left corner of the rectangle.

y: The y-coordinate of the top-left corner of the rectangle.

width: The width of the rectangle.

height: The height of the rectangle.

The strokeRect() method is used to draw the outline of a rectangle on the canvas.

4.Parameters of the canvas fillRect() method:

x: The x-coordinate of the top-left corner of the rectangle.

y: The y-coordinate of the top-left corner of the rectangle.

width: The width of the rectangle.

height: The height of the rectangle.

The fillRect() method is used to draw a filled rectangle on the canvas.

5.Ways to provide values to the fillStyle attribute of the canvas:

Color names: 'red', 'blue', 'green', etc.

Hexadecimal color codes: '#FF0000' for red, '#0000FF' for blue, etc.

RGB values: 'rgb(255, 0, 0)' for red, 'rgb(0, 0, 255)' for blue, etc.

RGBA values: 'rgba(255, 0, 0, 0.5)' for semi-transparent red, 'rgba(0, 0, 255, 0.5)' for semi-transparent blue, etc.

Gradient objects or patterns created using the createLinearGradient() or createPattern() methods.

6.Ways to pass an image to the drawImage() method of canvas:

Using an <img> element: context.drawImage(imgElement, x, y).

Using an Image object: context.drawImage(imageObject, x, y).

Using a canvas element: context.drawImage(canvasElement, x, y).

7.Methods to support text in canvas:

fillText(text, x, y): Draws filled text on the canvas at the specified coordinates.

strokeText(text, x, y): Draws the outline of the text on the canvas at the specified coordinates.

measureText(text): Returns a TextMetrics object containing information about the measured text.

8.Two canvas shadow properties and their purpose:

shadowColor: Sets the color of the shadow to be applied.

shadowBlur: Sets the blur level of the shadow.

9.The setInterval() method of canvas is not specific to canvas but a general JavaScript function used for executing a specified function repeatedly at a fixed time interval. It is often used in conjunction with canvas to create animations by repeatedly updating the canvas content within the specified interval.

10.The setTimeout() method of canvas is used to execute a specified function after a specified delay, executing it only once. It is often used in canvas applications to introduce a delay before performing a specific action or executing a particular code block.

4/6 Mark Questions:

1.Canvas methods explanation:

beginPath(): This method begins a new path or resets the current path. It is used to start defining a new shape or path on the canvas.

```
const canvas = document.getElementById('myCanvas');

const context = canvas.getContext('2d');

context.beginPath();
```

closePath(): This method closes the current path by drawing a straight line from the current point to the starting point of the path. It is used to close a shape or path, creating a closed figure.

```
context.closePath();
```

lineTo(x, y): This method adds a straight line segment to the current path, from the current point to the specified point (x, y).

```
context.lineTo(100, 100);
```

stroke(): This method strokes the current path with the current stroke style and line width, creating the visible outline of the path.

```
context.stroke();
```

2.Creating a linear gradient in canvas with a code example:

```
const canvas = document.getElementById('myCanvas');

const context = canvas.getContext('2d');

// Create a linear gradient

const gradient = context.createLinearGradient(0, 0, 200, 0);

gradient.addColorStop(0, 'red');

gradient.addColorStop(1, 'blue');

// Use the gradient as the fill style

context.fillStyle = gradient;

// Draw a rectangle filled with the linear gradient

context.fillRect(0, 0, 200, 200);
```

3.Creating a radial gradient in canvas with a code example:

```
const canvas = document.getElementById('myCanvas');

const context = canvas.getContext('2d');
```

```
// Create a radial gradient

const gradient = context.createRadialGradient(100, 100, 20, 100, 100, 100);

gradient.addColorStop(0, 'red');

gradient.addColorStop(1, 'blue');

// Use the gradient as the fill style

context.fillStyle = gradient;

// Draw a circle filled with the radial gradient

context.beginPath();

context.arc(100, 100, 100, 0, Math.PI * 2);

context.fill();
```

4.Explanation of arc() and arcTo() methods of canvas:

arc(x, y, radius, startAngle, endAngle, anticlockwise): This method adds an arc to the current path. It defines a circular or elliptical arc based on the provided parameters. The arc is drawn clockwise by default unless the anticlockwise parameter is set to true.

```
context.arc(100, 100, 50, 0, Math.PI * 2, false);
```

arcTo(x1, y1, x2, y2, radius): This method adds an arc that is tangent to the line segment connecting the current point to the point (x1, y1), and the line segment connecting (x1, y1) to (x2, y2). It creates a smooth curve between the two lines using the specified radius.

```
context.arcTo(100, 100, 200, 200, 50);
```

5.Explanation of quadraticCurveTo() and bezierCurveTo() methods of canvas with proper code example:

quadraticCurveTo(cp1x, cp1y, x, y): This method adds a quadratic Bézier curve to the current path. It defines a curve between the current point and the point (x, y), using the control point (cp1x, cp1y).

```
context.quadraticCurveTo(100, 100, 200, 200);
```

bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y): This method adds a cubic Bézier curve to the current path. It defines a curve between the current point and the point (x, y), using the control points (cp1x, cp1y) and (cp2x, cp2y).

context.bezierCurveTo(100, 100, 200, 200, 300, 300);

6.Explanation of scale(), rotate(), translate(), setTransform() methods of canvas using code examples:

scale(scaleX, scaleY): This method scales the current transformation matrix by the provided scaling factors. It resizes the canvas drawing accordingly.

context.scale(2, 2);

rotate(angle): This method rotates the current transformation matrix by the specified angle in radians. It rotates the canvas drawing around the origin.

context.rotate(Math.PI / 4);

translate(x, y): This method translates the current transformation matrix by the given offsets (x, y). It moves the canvas drawing by the specified amount.

context.translate(100, 100);

setTransform(a, b, c, d, e, f): This method sets the current transformation matrix to the specified values. It replaces the existing transformation matrix with a new one defined by the matrix elements (a, b, c, d, e, f).

context.setTransform(1, 0, 0, 1, 100, 100);

7.Canvas shadow properties:

shadowColor: Sets the color of the shadow. It accepts a CSS color value.

shadowBlur: Sets the blur level of the shadow. It defines the size of the blur effect.

shadowOffsetX: Sets the horizontal offset of the shadow from the object.

shadowOffsetY: Sets the vertical offset of the shadow from the object.

Example usage:

context.shadowColor = 'rgba(0, 0, 0, 0.5)';

context.shadowBlur = 10;

context.shadowOffsetX = 5;

context.shadowOffsetY = 5;

8.Explanation of lineCap and lineJoin attributes of canvas with all their possible values:

lineCap: Specifies the shape used to draw the ends of lines. Possible values are:

butt (default): Ends the line with a flat edge.

round: Ends the line with a rounded edge.

square: Ends the line with a square edge.

lineJoin: Specifies the shape used to join two lines where they meet. Possible values are:

miter (default): Joins the lines with a sharp corner.

round: Joins the lines with a rounded corner.

bevel: Joins the lines with a beveled corner.

Example usage:

context.lineCap = 'round';

context.lineJoin = 'bevel';

9.Canvas compositing options (globalCompositeOperation attribute values):

source-over (default): Displays the source image on top of the destination image.

source-in: Displays the source image where it overlaps with the destination image, and makes the rest transparent.

source-out: Displays the source image where it does not overlap with the destination image, and makes the rest transparent.

source-atop: Displays the source image on top of the destination image, but only where they overlap.

destination-over: Displays the destination image on top of the source image.

destination-in: Displays the destination image where it overlaps with the source image, and makes the rest transparent.

destination-out: Displays the destination image where it does not overlap with the source image, and makes the rest transparent.

destination-atop: Displays the destination image on top of the source image, but only where they overlap.

lighter: Displays the sum of the source image and the destination image.

copy: Displays only the source image, ignoring the destination image.

xor: Displays the source image where it overlaps with the destination image, and the destination image where it does not overlap with the source image.

Example usage:

context.globalCompositeOperation = 'source-in';

10.Steps to implement canvas animation:

Set up the canvas element in your HTML:

<canvas id="myCanvas" width="400" height="400"></canvas>

Get the canvas drawing context in JavaScript:

const canvas = document.getElementById('myCanvas');

const context = canvas.getContext('2d');

Define the initial state of your drawing.

Use the requestAnimationFrame method to create a loop that updates the canvas in regular intervals:

function animate() {

// Update the canvas drawing here

// Call animate() recursively to create a loop

requestAnimationFrame(animate);

}

// Start the animation

animate();

Inside the animate function, update the canvas drawing based on the desired animation effect.

Use methods such as clearRect to clear the canvas or drawImage to draw images in each frame of the animation.

Use techniques like changing the position, scale, or rotation of objects, and modifying attributes such as colors or opacity, to create the desired animation effect.

11.Explanation of setInterval(), setTimeout(), and requestAnimationFrame() methods with their parameters:

setInterval(callback, delay): This method repeatedly calls the callback function at the specified delay interval (in milliseconds) until cleared or stopped. It is commonly used to create a continuous animation loop.

constintervalId = setInterval(callback, 1000);

setTimeout(callback, delay): This method calls the callback function after the specified delay time (in milliseconds) has passed. It is commonly used to perform a task after a certain delay.

consttimeoutId = setTimeout(callback, 2000);

requestAnimationFrame(callback): This method requests the browser to call the callback function before the next repaint. It is optimized for animations and provides better performance compared to setInterval or setTimeout for smooth animations.

```
function animate() {
  // Update canvas animation here
  // Call requestAnimationFrame() recursively to create a loop
requestAnimationFrame(animate);
}
// Start the animation
requestAnimationFrame(animate);
```

These methods are commonly used in canvas animations to control the timing and execution of animation frames.

****